Ad Hoc Networks 10 (2012) 1–18

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Neighbor discovery in wireless networks with sectored antennas ☆

R. Murawski [a,*], E. Felemban [b,1], E. Ekici [a], S. Park [c], S. Yoo [c], K. Lee [c], J. Park [c], Z. Hameed Mir [c]

[a] Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA
[b] Department of Computer Engineering, Umm Al Qura University (UQU), Makkah, Saudi Arabia
[c] Electronics and Telecommunication Research Institute (ETRI), Daejeon, South Korea

ARTICLE INFO

Article history:
Received 8 October 2010
Received in revised form 17 March 2011
Accepted 3 April 2011
Available online 13 April 2011

Keywords:
Wireless ad hoc networks
Sectored antennas
Directional antennas
Neighbor discovery

ABSTRACT

Directional antennas offer many potential advantages for wireless networks such as increased network capacity, extended transmission range and reduced energy consumption. Exploiting these advantages requires new protocols and mechanisms at various communication layers to intelligently control the directional antenna system. With directional antennas, many trivial mechanisms, such as neighbor discovery, become challenging since communicating parties must agree on where and when to point their directional beams to communicate.

In this paper, we propose a fully directional neighbor discovery protocol called Sectored-Antenna Neighbor Discovery (SAND) protocol. SAND is designed for sectored-antennas, a low-cost and simple realization of directional antennas, that utilize multiple limited beamwidth antennas. Unlike many proposed directional neighbor discovery protocols, SAND depends neither on omnidirectional antennas nor on time synchronization. SAND performs neighbor discovery in a serialized fashion allowing individual nodes to discover all potential neighbors within a predetermined time. SAND guarantees the discovery of the best sector combination at both ends of a link, resulting in more robust and higher quality links between nodes. Finally, SAND reliably gathers the neighborhood information in a centralized location, if needed, to be used by centralized networking protocols. The effectiveness of SAND has been assessed via simulation studies and real hardware implementation.

1. Introduction

Performance improvement through directional antennas in wireless networks has been discussed extensively in the literature [1–4]. Directional antennas, as opposed to omnidirectional antennas, concentrate the transmission power towards a certain direction with limited beamwidth around this direction. As a result, directional antennas lend many promising features to wireless networks. First, interference between neighboring nodes is greatly reduced which increases the simultaneous transmissions in the neighborhood and network capacity [5]. Second, signal to interference and noise ratio (SINR) is increased due to higher directional gain allowing transmissions a higher data rates. Finally, for a given transmission power, the communication range is greatly extended. Alternatively, lower transmission power is needed to cover the same transmission range covered by an omnidirectional antenna.

One of the simplest realizations of directional antennas is sectored antennas where a fixed number of fixed beamwidth antenna elements are mounted to cover the whole azimuth. One major advantage of sectored antennas compared to other types of directional antennas is its low-cost and implementation simplicity. Switching sectors is done by simply selecting an antenna element as shown in Fig. 1.

☆ A preliminary version of this paper has appeared in the proceedings of IEEE SECON 2010.

* Corresponding author. Tel.: +1 321 947 5188.
E-mail addresses: murawskr@ece.osu.edu (R. Murawski), eafelemban@uqu.edu.sa (E. Felemban), ekici@ece.osu.edu (E. Ekici), sangjoon@etri.re.kr (S. Park), yoos@etri.re.kr (S. Yoo).

[1] This work has been done while Emad Felemban was at The Ohio State University.

1570-8705/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2011.04.001

Protocol design for wireless networks with directional antennas is a challenging problem due to the problems related to directional antennas such as "Directional Hidden Terminal" problem [6] and the "Deafness" problem [7]. In addition to these problems, basic network operations such as neighbor discovery become more complicated, as well.

Neighborhood information plays an important role in multihop wireless networks for routing, clustering and MAC operation. Neighbor discovery is a relatively simpler problem when omnidirectional antennas are used since a simple broadcast can reach all nodes within the transmission range. The problem, however, becomes more challenging when directional antennas are used due to the following reasons: (i) The limited radial range of the beamwidth of the directional antenna that covers only a fraction of the azimuth. This limitation requires the neighbor discovery scheme to be repeated in different directions to cover the whole azimuth. (ii) Neighboring nodes must know when and where to point their directional beams to discover each other. (iii) Due to non-ideal realizations of directional antennas (i.e., the existence of side lobes), two neighboring nodes might find multiple links between themselves through different Sector-to-Sector (S2S) links.

Many neighbor discovery protocols have been proposed for wireless networks that use directional antennas. The main objective of the these protocols is to discover the neighbors around a node and store the neighborhood information locally. In the literature, three main approaches were used to perform the neighbor discovery. A set of proposed protocols utilize an omnidirectional antenna to bootstrap the neighbor discovery process [8–10]. Other protocols require time synchronization to perform neighbor discovery [8,11,12]. Finally, random schemes [13] perform the neighbor discovery by sending Hello packets through different random directions and then listening to another random direction. As discussed in Section 2, all these approaches are associated with significant shortcomings.

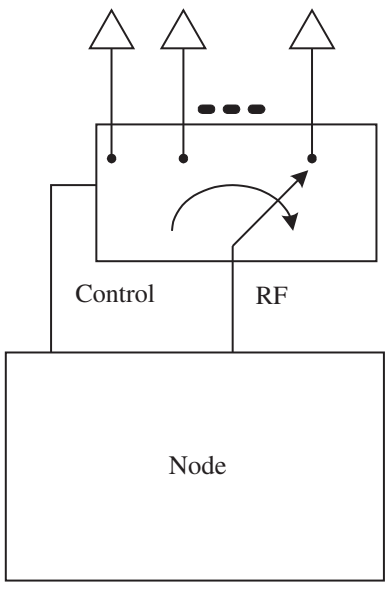


Fig. 1. Node architecture with sectored antennas.

In this paper, we introduce the Sectored-Antenna Neighbor Discovery (SAND) protocol, an integrated neighbor discovery mechanism that allows individual nodes to discover their neighbors and report their neighborhood information back to a centralized location for later use. The key idea of our proposed protocol is to serialize the neighbor discovery process among all nodes. This allows each node to perform neighbor discovery in a bounded time and reliably relay the neighborhood information. Unlike many other directional neighbor discovery protocols, SAND does not require an omnidirectional antenna or external global synchronization. Moreover, SAND discovers all S2S links between nodes that they can communicate over using a systematic approach. SAND can be utilized with any distributed or centralized routing or MAC protocol that uses directional antennas since neighborhood information is available both centrally and locally.

2. Related work

The performance improvements through directional antennas are well explored in the literature [1–4] for ad hoc wireless networks, mesh networks [14] and sensor networks [15,10]. To exploit these advantages many protocols have been proposed for MAC layer [3,16–18], routing [19,20] and topology control [21,22]. Several neighbor discovery schemes have been proposed in the literature, as well. Some of the schemes utilize an omnidirectional antenna to bootstrap the neighbor discovery process like in [8–10]. Utilizing a secondary omnidirectional antenna for neighbor discovery has several disadvantages. First, differences in the omnidirectional and sector antenna gain patterns could result in different sets of discovered neighbors. Second, when discovering neighbors in SAND, we are not only interested in the identity of potential neighbors, but in measuring the link quality for the various Sector-to-Sector (S2S) links. This additional consideration cannot be realized using omnidirectional antennas. Finally, the cost of a secondary antenna system, along with the associated hardware required to support it, is undesirable in cost-sensitive networks. These schemes also assume that the discovering node knows its geographical location and can include this information in the advertisement packets which may be an expensive option for some wireless network implementations.

To eliminate the need for omnidirectional antennas, other proposed neighbor discovery schemes assume that nodes are time synchronized to guarantee that all nodes switch their sectors synchronously such as in [11,13,23]. While time synchronized neighbor discovery schemes eliminate the usage of omnidirectional antennas, they require time synchronization for all nodes which may increase the hardware and control overhead burden on the wireless network. Furthermore, achieving time synchronization in a distributed manner is also very unlikely without an initial communication infrastructure and solely based on directional/sectored antennas.

Asynchronous and fully directional neighbor schemes were proposed in [2,13] where each node listens in a random direction for a random time and then transmits a

hello message in a random direction and goes back to listening in another random direction. Although this approach is simple and straightforward, it does not bound the time for neighbor discovery since it is a probabilistic approach. Moreover, it does not guarantee that discovered links are bidirectional.

With SAND protocol, directional neighbor discovery is done solely using directional antennas in both receiver and transmitter, allowing the discovery of more nodes than when using omnidirectional antennas. Moreover, SAND protocol does not require each node to know its geographical location or assume time synchronization between all nodes. Only local synchronization is required when exchanging initial control messages, and this limited time synchronization is handled solely by the SAND protocol. The serialized approach of SAND allow nodes to discover all potential Sector-to-Sector (S2S) links between a given pair of nodes which has been completely disregarded in the literature. In addition to discovering neighbor nodes and storing the neighborhood information locally, SAND gathers the neighborhood information from all nodes in a centralized location to be used, if needed, by centralized networking protocols. To the best of our knowledge, SAND is the first neighbor discovery protocol that has all of these features.

3. Sectored antenna neighbor discovery

SAND is designed for a K sectored antenna that covers the entire omnidirectional 360° range, as shown in Fig. 2. SAND can handle non-ideal, overlapping sectored-antennas coverage. Switching between sectors is done by simply selecting a different antenna element, as shown in Fig. 1, allowing this type of directional antenna to be applicable for a wider range of wireless network types. Each node can only activate one sector at a time for both transmission and reception. In this system, SAND does not require an additional omnidirectional antenna.

The key idea of SAND is the serialization of the neighbor discovery process among all nodes. While serialization results in a linear increase in discovery time with network size, it is important to note that neighbor discovery is performed at network initialization and for infrequent network topology updates. The concepts of SAND can be extended to more frequent, incremental updates as well. However, this is beyond the scope of this paper.

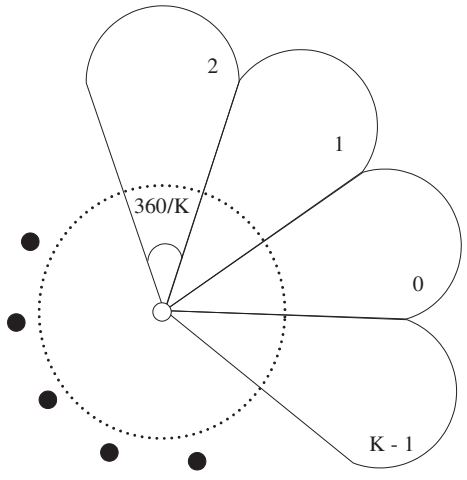


Fig. 2. Idealized K sectored-antenna.

Serialized neighbor discovery allows each node to perform neighbor discovery within a bounded time. Moreover, each node can relay its neighborhood information to a centralized location reliably. Serialization is achieved using a token passing approach where only the node that holds the token (Token-Holder node) is allowed to perform the neighbor discovery. The neighbor discovery token (ND-Token) is centrally controlled by the node which will gather and process all neighborhood information, such as a cluster head or sink node. Centralized control of the ND-Token has several benefits. First, by directly controlling where the ND-Token is sent, the central controller can estimate when the ND-Token should be returned. If the ND-Token is then lost, the central controller can generate a new ND-Token and recover the neighbor discovery process. Second, the information gathered on nodes can be returned, if needed, along with the ND-Token, to the centralized controller [24].

Once a node receives the ND-Token, the neighbor discovery process starts which can be summarized in three steps:

1. Holding the attention of its neighbors (Honing In mechanism).
2. Discovering neighbors (Hello-Reply mechanism).
3. Returning the token to the central neighbor discovery controller (Token Passing and Releasing Mechanisms).

For SAND to operate, we assume that the number of sectors, K, for each node is equal. Also, we assume that a single sector is active at one time, and that the time required to switch the active sector is negligable. Finally, we assume that the control parameters presented in the following sections are known by all nodes a-priori.

3.1. Initialization

Initially, nodes start in *Fast-Scan* mode where they continually switch their active sector sequentially looking for any channel activity in the active sector. During this mode, each node waits for t_{switch}s before switching to another sector. Note that, such switching is not synchronized among nodes. The ND-Token is originally assigned to the central neighbor discovery controller. The node that possesses the ND-Token is referred to as the Token-Holder.

3.2. Hone-In mechanism

Before the Token-Holder can discover its neighbors, the Token-Holder must hold the 'attention' of its neighbors. Recall that, initially, all nodes are in the *Fast-Scan* mode, switching their sectors scanning for any channel activity. The Token-Holder begins the Hone-In process by broadcasting a series of N_{HoneIn} *Hone-In* beacons to each sector in series. The *Hone-In* beacons are sent periodically every

t_{HoneIn} seconds. Each beacon message contains a sequence number, x, which indicates the number of remaining *Hone-In* beacons that will be sent before the Token-Holder is finished with the Hone-In process.

To guarantee that all local neighbor nodes have an opportunity to receive the beacon message, the duration of the periodic beacon transmissions in each sector must be sufficiently long that one full round of sector scanning by the neighbor nodes can occur. We assume there is an arbitrarily small delay in the sector switching of the neighbor nodes, $t_{switchDelay}$, which accounts for the hardware switching time, as well as any discrepancies in the local timers of the Token-Holder and neighbor nodes. Therefore the time required for a complete scan of all sectors by a given neighbor node is shown in Eq. 1,

$$T_{scan} = (t_{switch} + t_{swithDelay}) \times K, \tag{1}$$

where K is the number of sectors for all nodes.

To ensure that all neighbors are able to receive at least one *Hone In* message, the number of beacons transmitted must satisfy the following,

$$N_{HoneIn} > \frac{(t_{switch} + t_{switchDelay}) \times K}{t_{HoneIn}}. \tag{2}$$

To simplify this equation, we assume that the duration of the switching delay, $t_{switchDelay}$, is sufficiently short that K delays are less than the duration of one Hone-In periodic transmission. Therefore, the number of *Hone-In* messages required is

$$N_{HoneIn} = \frac{t_{switch} \times K}{t_{HoneIn}} + 1. \tag{3}$$

In the implementation and simulation of the SAND protocol, we assume that the duration a neighbor node scans a given sector, t_{switch} is twice that of the periodic *Hone-In* broadcast delay, t_{HoneIn}. Therefore, in the following sections, we set N_{HoneIn} to be $2K + 1$.

An example of the timing associated with the Hone-In process is shown in Fig. 5. In this example, two neighbors receive the beacon message from the Token-Holder node. Sector 2 of Neighbor A is facing sector 2 of the Token-Holder while sector 1 of Neighbor B is facing sector 1 of the Token-Holder. When the sectors are aligned properly, the neighbor node ceases switching of sectors and starts a timer which will expire when the Token-Holder node has complete its Hone-In process. Note that, the *Hone-In* mechanism initiates local time synchronization for the remaining of the neighbor discovery of the Token-Holder node. The time synchronization is not needed once token is passed.

The time T_{HoneIn} required for the hone is

$$T_{HoneIn} = N_{HoneIn} \times t_{HoneIn} \times K, \tag{4}$$

After the Hone-In process is complete, the Hello-Reply timer will expire on all neighboring nodes, indicating that the Hello-Reply process should start. A complete example of the SAND Hone-in process is illustrated in Fig. 3.

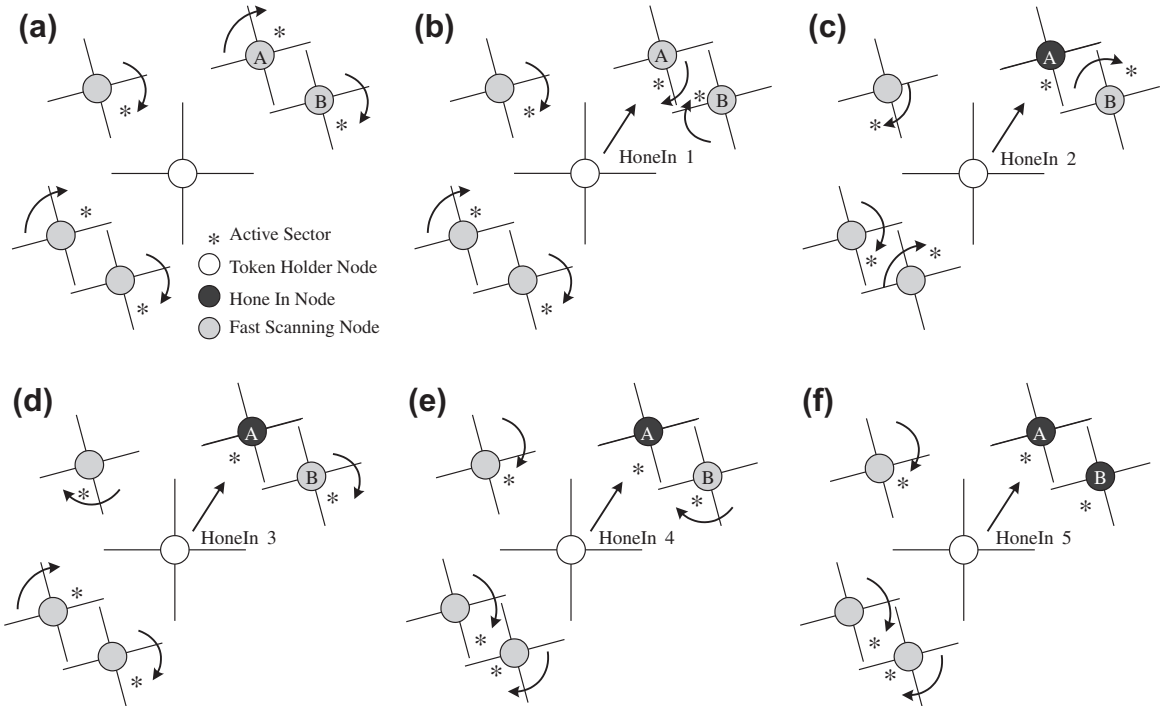


Fig. 3. Hone-In process with four sectored antenna: (a) all nodes except the Token-Holder node are in Fast Scan mode, (b) Token-Holder broadcasts the first Hone-In beacon to its first sector while node *A* activates the sector pointing towards the Token-Holder node, (c) Node *A* receives the Hone-In beacon and locks its sector while node *B* continues switching its sectors, (d) Token-Holder broadcasts its third Hone-In beacon, (e) Node *B* switches to the sector pointing toward the Token-Holder node and receives the 4th Hone-In beacon, (f) Node *B* locks its sector towards the Token-Holder node and stops switching its sectors.

3.3. Hello-Reply mechanism

After the Hone-In process completes, all neighbor nodes start the Hello-Reply process by switching to the first sector. The Token-Holder node then begins discovering its neighbors by polling the neighbors and waiting for their replies. This two-way handshake guarantees the discovery of bidirectional links to all neighbor nodes. During the Hello-Reply process, both the Token-Holder and neighbor nodes loops through their K sectors in a pre-determined time sequence, ensuring that the communication over all S2S links are tested for later comparison. The process of sequentially testing all sector combinations is illustrated in Fig. 4.

For all K^2 S2S links, the Token-Holder broadcasts N_{rounds} *Hello* packets, waiting for *Reply* packets from neighbors after each transmission. Neighbor nodes, after receiving the *Hello* packet, select a random slot for their response. A maximum of N_{slots} are available after each *Hello* message. The slotted reply mechanism is intended to reduce the probability of collision between neighbor nodes. If two nodes do select the same slot, their *Reply* packets will collide and thus will not be discovered by the Token-Holder in this Hello-Reply round. Each *Reply* packet contains the ID and sector of the originating node. Also, the *Reply* packet contains an bit flag to indicate whether or not the neighbor node has performed its own neighbor discovery yet. After receiving the *Reply* packet, the Token-Holder adds the neighbor's information into its locally collected neighbor table.

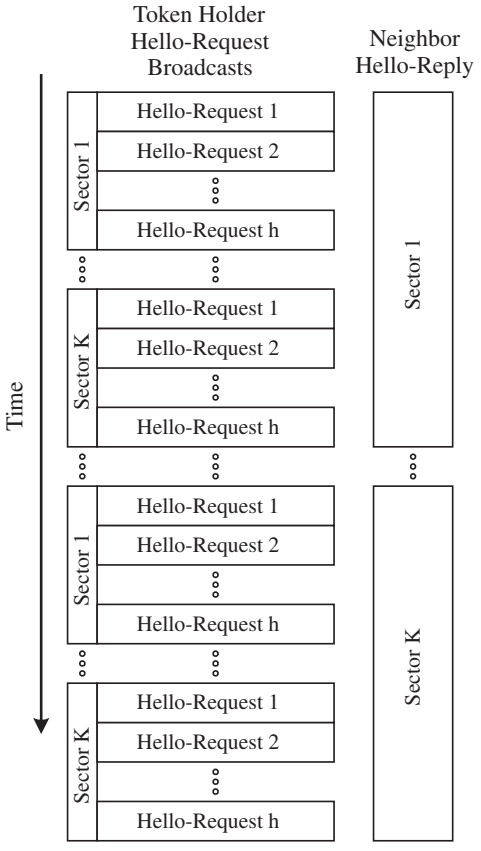


Fig. 4. Sector Switching for Hello-Reply Mechanism

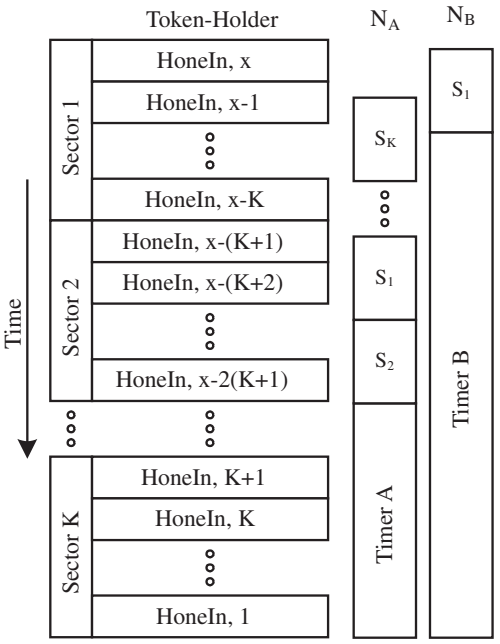


Fig. 5. Hone-In Process for Two Neighbor Nodes.

The Token-Holder repeats *Hello-Reply* packet exchange for N_{rounds} times for each sector-pair to discover new neighbors which their *Reply* packets have collided in previous exchanges. To maximize the probability of discovering all nodes within a bounded time, the Token-Holder node transmits the IDs of all neighboring nodes that have been discovered using a Long Hello packet in successive *Hello-Reply* rounds. Only nodes that do not read their IDs in this last packet are allowed to reply. This mechanism allows unlucky nodes that have not been discovered yet due to *Reply* packet collisions to send a successful *Reply* packet with a higher probability.

An example of the *Hello-Reply* packet exchange for one sector pair is given in Fig. 6. After neighbor nodes receive the first *Hello* packet, they start a slotted time and select a random slot to transmit their *Reply* packets. Both node B and D select the slot number 1 to send their replies. As a result their reply packets collide, while nodes A and C select the 0th and last slots, respectively. After the first *Hello-Reply* round, the Token-Holder discovers both nodes A and C but not B and D. In the second *Hello-Reply* round, the Token-Holder includes the IDs of the already discovered nodes, nodes A and C, in the Hello packet. Allowing more chance for both nodes B and D to be discovered.

This *Hello-Reply* packet exchange is repeated in the remaining $K-1$ sectors. Obviously, the time $T_{HelloReply}$ needed to finish this process depends on the number of slots N_{slots} and the number of *Hello-Reply* packets rounds N_{rounds} and can be calculated as:

$$T_{HelloReply} = K^2[(\tau_{Hello} + \tau_{HelloLong}(N_{rounds} - 1) + N_{slots}\tau_{ReplySlot}N_{rounds}], \quad (5)$$

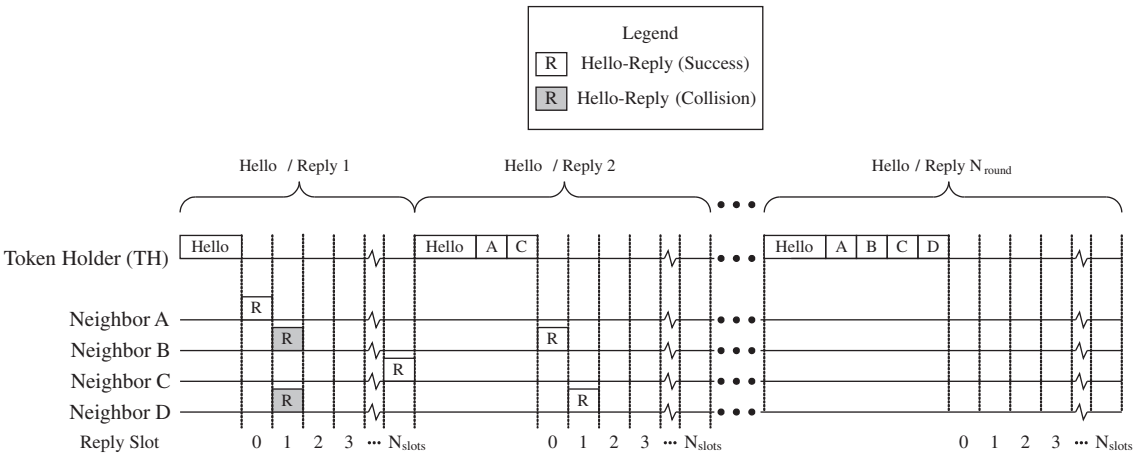


Fig. 6. Hello-Reply Messages Exchange.

where τ_{Hello} is the transmission length of a Hello packet, $\tau_{HelloLong}$ is the average transmission length of the Long Hello packets, and $\tau_{ReplySlot}$ is the duration of one *Hello-Reply* slot.

3.4. Token passing mechanism

After the Token-Holder finishes its local directional neighbor discovery mechanism as described in Section 3.3, the ND-Token must be handed to another node to continue the network neighbor discovery process. Recall that, initially, the central neighbor discovery controller must discover its direct neighbor nodes to kick off the network neighbor discovery. The direct neighbors of the central node are finished with the *Hello-Reply* mode when they have scanned all K sectors, after which they return to *Fast-Scan* mode. The Token-Holder's local neighbor discovery process is complete after all K sectors have been scanned K times, as described in Section 3.3. After completing its local neighbor discovery, the central node waits for $t_{TokenHandle}$ seconds before entering the *Token-Handling* state. Due to the node synchronization described in Section 3.2 all nodes will finish neighbor discovery within a small margin of error. By properly setting $t_{TokenHandle}$ to account for such synchronization discrepancies, the central node will enter the *Token-Handling* state after all neighbors have successfully entered the *Fast-Scan* mode. In the *Token-Handling* state, the central node passes the ND-Token sequentially to all nodes that have not yet discovered their immediate neighbors as described in the following section.

Recall that, during the exchange of neighbor information described in Section 3.3, a bit is set in the *Hello-Reply* message which indicates whether the replying node has performed its neighbor discovery or not. This information is stored in the local neighbor table at each Token-Holder, and is passed to the central node during *Token-Release* as described in the following section. In *Token-Handling* mode, the central node scans its neighbor table for the first node which has not yet performed neighbor discovery, this node will become the next Token-Holder. Before sending the neighbor-discovery token to the next Token-Holder node, however, the central node must first discover a route to the next Token-Holder node. Given that the central node has collected all neighbor information from the previous Token-Holders, all information required to form a temporary route to the next Token-Holder is stored at the central node. In our implementation, we utilize a shortest-path algorithm to generate the temporary routing table used for neighbor discovery transmissions. However, based on what information is gathered, more advanced route discovery algorithms could be used. As the temporary routing tree is only stored at the central node, the routing information must be embedded into any multi-hop transmission so that intermediate nodes can forward the ND-Token to its proper destination.

The ND-Token is sent through the discovered end-to-end route from the central node to the new Token-Holder using reliable acknowledged unicast transmissions. In the case that an acknowledged token transmission fails, due to link or node failure, the source of the transmission will attempt $N_{retransmission}$ retransmissions before returning the ND-Token to the central node and indicating a link-failure has been detected. Before transmitting any message, the source node must catch the attention of its destination neighbor. This is accomplished utilizing a simplified version of the broadcast Hone-In mechanism described in Section 3.2. Instead of broadcasting *Hone-In* messages to all K sectors, unicast *Hone-In* messages are sent only to the sector used for communication with the next-hop node. The unicast *Hone-In* message can contain the ID of the destination node to ensure that only the intended recipient leaves the *Fast-Scan* mode. Because the next-hop node is in the *Fast-Scan* mode, the same settings used for the individual sectors in Section 3.2 can be used in this *Hone-In* mechanism. After transmitting the neighbor-discovery token, the transmitting node must return to *Fast-Scan* mode to scan for future activity. A complete example of the SAND Token-Passing mechanism is shown in Fig. 7.

3.5. Token releasing mechanism

When a node finished the discovery of its neighbors, the neighbor-discovery token and all collected neighbor information must be returned to the central node for process-

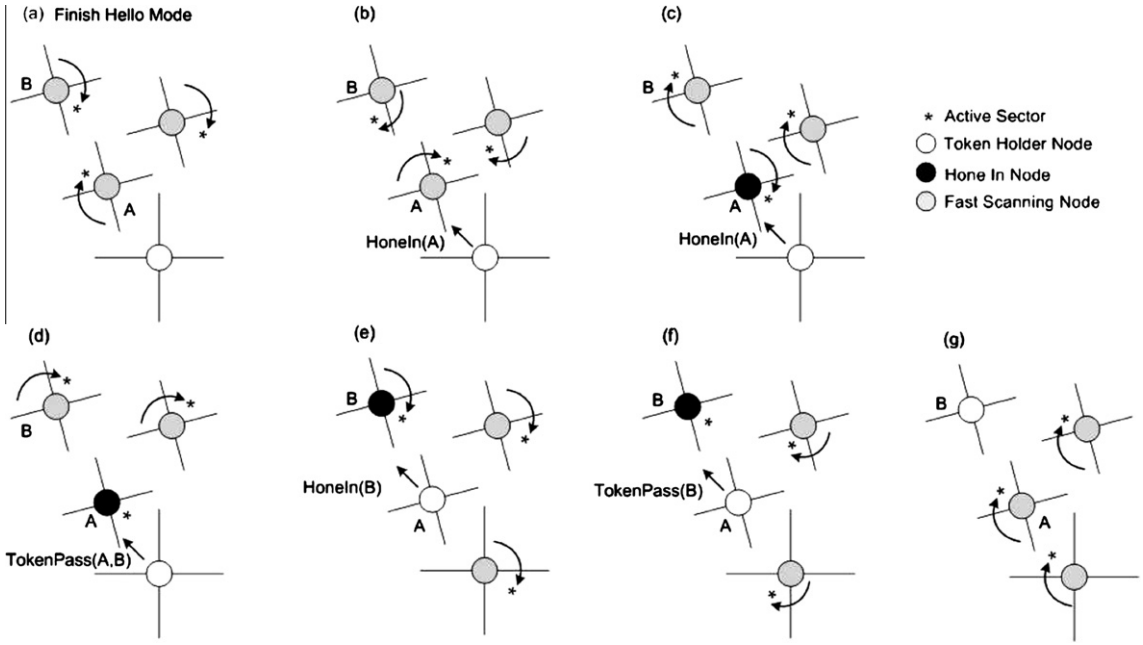


Fig. 7. Token Passing Mechanism: (a) Following the *Hello-Reply* mode, all neighbors enter *Fast-Scan* state, (b) The Token-Holder sends the unicast *Hone-In* beacons in the direction of node *A*, (c) When node *A* activates the proper sector for communication with the Token-Holder, it leaves *Fast-Scan* mode, awaiting a control message from the Token-Holder, (d) The Token-Holder sends an acknowledged control packet which includes the ID of the new Token-Holder (*B*) and all routing information for node *A* to forward the ND-Token, (e) Node *A* sends the unicast *Hone-In* beacons in the direction of node *B*, Node *B* leaves *Fast-Scan* mode, awaiting a control message from node *A*, (f) Node *A* sends an acknowledged control packet indicating that node *B* is the new Token-Holder, (g) all nodes except the new Token-Holder (node *B*) are in fast scan mode, node *B* is ready to begin local neighbor discovery.

ing. The same unicast *Hone-In* process can be used as is described in Section 3.4 to transmit the *Token-Release* message as well as all gathered neighbor information. For nodes that are multiple hops away from the central node, the routing information that was used to originally transmit the token can be used in reverse to transmit the data to the central node. After all neighbors have received the neighbor-discovery token, and have successfully released their collected neighbor information to the central node, neighbor discovery is complete.

3.6. Token recovery mechanism

In previous versions of the SAND protocol [24] a distributed version of token-passing was utilized in which local decisions made by the current Token-Holder dictated to which node the neighbor-discovery token would go next. This allowed the Token-Pass mechanism to naturally generate a depth-first routing tree, which was utilized for the collection of neighborhood information. However, due to this distributed token-passing mechanism, when a neighbor discovery token is transmitted, we cannot reasonably know how long it will would be until the token should be returned. Given a highly connected network, it is possible that after the token leaves the first node, the entire network would be discovered before the token is returned. If we make no assumptions on the size of the network prior to neighbor discovery, this leads to several problems in ensuring the reliability of the token-passing mechanism.

There are many errors that could lead to the loss of a neighbor-discovery token, including node failure, unreliable communication, etc. If the token is lost, four steps must be taken to recover and continue with the neighbor discovery process:

1. Detect the token loss.
2. Report the loss to the central node.
3. Adjust the current neighbor table to reflect the loss.
4. Generate a new token.

Given the centralized approach to Token-Passing as described in Section 3.4 detection of a lost token is quite trivial. When a new Token-Holder is selected at the central node, all routing information in known at the central node. This routing information is also embedded in the token-passing message, so all intermediate nodes can know the number of hops left in the multi-hop token-pass message. The time required for each hop of the route to Hone-In the neighbor and reliably transmit the token is known. Also, once the token arrives at the pre-determined Token-Holder, the time required for the Hone-In and hello message exchange is known. The only unknown variable is the number of neighbors the new Token-Holder will discover. As this number directly affects both the size and the transmission time for the *Token-Release* message, a reasonable amount of time must be allotted as to not prematurely declare a token as lost. At the central node, as well as each intermediate node, a timer is set based for when the token should be returned. If this timer expires, each node

in the token-route will transmit a special control message to the central node using the unicast Hone-In mechanisms described in Section 3.4.

After detecting the link error, the central node can update neighbor table and temporary routing table to avoid the bad link. The next Token-Holder can then be selected in the same manner as described in Section 3.4.

3.7. Token-Recovery example

To demonstrate the importance of token recovery in a practical network, we present the basic network topology shown in Fig. 8. As the ND-Token is passed around the network (Figs. 8A, B) using the centralized token passing mechanism as described in Section 3.4, the gathered neighbor data is collected at the central controller, Node 1. In Fig. 8.C, after the token is passed to Node 4, we disable communication between Node 2 and Node 4 to simulate outside interference making communication impossible. Therefore, when Node 4 is finished with its local neighbor discovery, it will fail in its attempts to send the token and its locally collected neighbor data to Node 1. Without a mechanism to recover from the loss of the neighbor discovery token, the neighbor discovery process would fail at this point, resulting in a partially undiscovered network. In the case of SAND, the loss of this connection is discovered in several places. Node 4 discovers the link failure after $N_{retransmission}$ unsuccessful transmission attempts to Node 2. Node 2 and Node 1 both discover the link failure when their respective token return timers expire. After the link failure is discovered, Node 4 simply returns to the *fast-scan* mode, awaiting another token. Node 2 informs Node 1 of the specific link failure that caused the token loss. Node 1, removes the bad link from the neighbor table and continues with the SAND token passing process while avoiding the bad link (Fig. 8.D). In realistic network environments, link failures such as this cannot be completely avoided, however, by implementing this recovery mechanism the neighbor discovery process can remain functional.

4. Analysis of Hello-Reply mechanism

In this section, we present an analytical model to analyze the Hello-Reply mechanism of SAND. Furthermore, an optimization algorithm is proposed to select optimal parameter configuration of the Hello-Reply mechanism (i.e. the number of time slots N_{slots} and the number of rounds N_{rounds}). For convenient notation, we use s to refer to the number of slots N_{slots} and r to refer to the number of rounds N_{rounds}.

4.1. Analytical model for slot selection mechanism

To discover a node during an arbitrary slot time 'a' out of s slots, exactly one node out of n neighboring

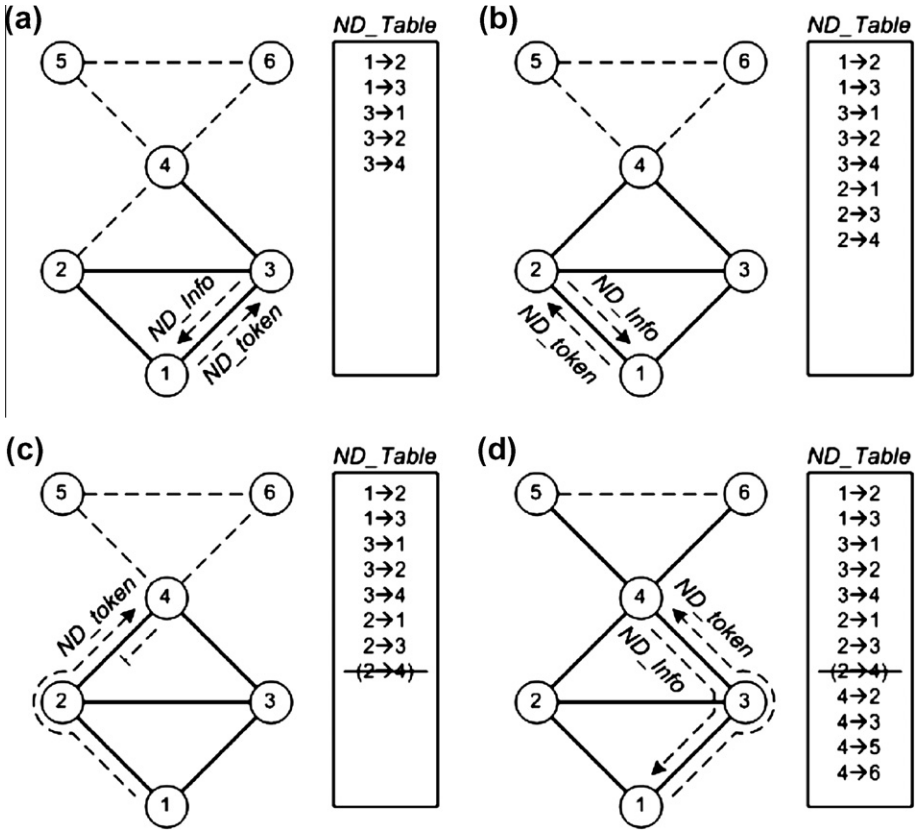


Fig. 8. Neighbor discovery with token recovery.

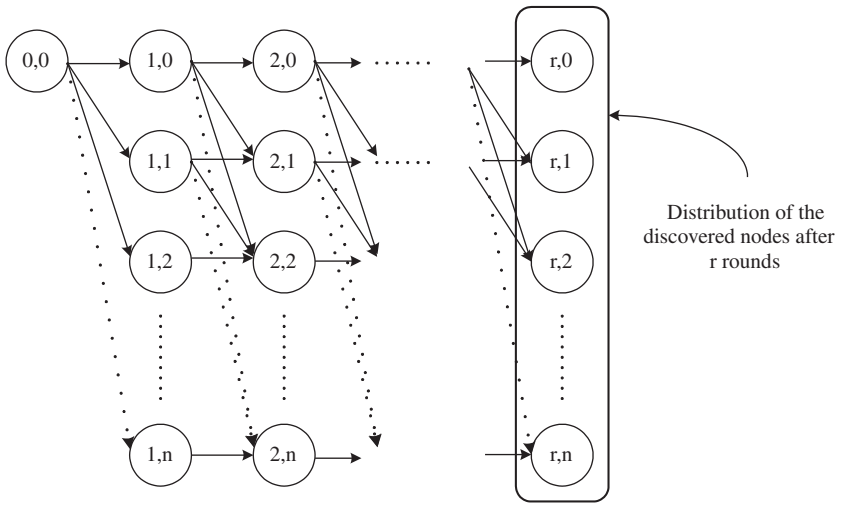


Fig. 9. Two dimensional map for r rounds and n nodes.

nodes should select that slot and transmit its reply packet.[2] Let us define slot 'a' as a **Discovery Slot**. Then, the probability $p(n,s)$ that an arbitrary slot is a discovery slot (i.e. exactly one node transmits during that slot) can be given as

$$p(n,s) = \binom{n}{1}\left(\frac{1}{s}\right)\left(1-\frac{1}{s}\right)^{n-1}, \tag{6}$$

where $\frac{1}{s}$ is the probability that a node selects a random slot out of s slots. Let $m(n,s)$ be the maximum number of nodes that can be discovered during a single Hello-Reply round given that there are n nodes competing for s slots. Formally, $m(n,s)$ can be defined as

$$m(n,s) = \begin{cases} s-1 & \text{if } n > s \\ n & \text{if } n \leqslant s \end{cases}. \tag{7}$$

Then, the probability $q_1(n,s)$ of discovering exactly 1 node out of n nodes during a Hello-Reply round that consists of s slots is given as

$$q_1(n,s) = \binom{s}{1} p(n,s) l(n-1,s-1), \tag{8}$$

where $l(n-1,s-1)$ is the probability that none of the remaining $(s-1)$ slots is a discovery slot. Alternatively, $l(n-1,s-1)$ can be defined also as the probability that none of the remaining $(n-1)$ nodes is discovered during the remaining $(s-1)$ slots. The term $p(n,s)$ is the probability of having an arbitrary discovery slot and $\binom{s}{1}$ is the number of ways this arbitrary slot can be selected. The probability $q_i(n,s)$ of discovering i nodes $(1 \leqslant i \leqslant m(n,s))$ in one Hello-Reply round can be calculated as

$$q_i(n,s) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n \leqslant s \text{ and } i = n-1 \\ \binom{s}{i}\left[\prod_{j=0}^{i-1} p(n-j,s-j)\right] & \\ \times l(n-i,s-i) & \text{Otherwise} \end{cases}. \tag{9}$$

which can be used to calculate the probability mass function of the number of neighbor nodes that can be discovered in one Hello-Reply round.

Finally, the probability $l(n-i,s-i)$ is defined as the probability that none of the remaining $(s-i)$ slots are discovery slots, or alternatively, it is the probability that no discovery slot exist among $(s-i)$ slots and is calculated as

$$\begin{aligned} l(n-i,s-i) &= q_0(n-i,s-i) \\ &= 1 - \sum_{k=1}^{m(n-i,s-i)} q_k(n-i,s-i). \end{aligned} \tag{10}$$

4.2. Analytical model for Hello-Reply rounds

Recall that from Eq. 7, the discovering node can only discover up to $m(n,s)$ nodes in one Hello-Reply round. To estimate the probability that k nodes are discovered when the neighbor discovery process ends (i.e., after r Hello-Reply rounds), we must keep track of different possibilities that lead to discovering the k nodes in r rounds.

To account for such possibilities, multiple Hello-Reply rounds can be described as a two dimensional probability map with $(n+1) \times r$ states as shown in Fig. 9. Each state (x,y) represents the probability that y nodes are discovered up to and including the xth round. Any state (x,y) can only access states $(x+1,y)$, $(x+1,y+1)$, ... or $(x+1,\min(y+m(y,s), n))$. The transition probabilities from states (x,a) to $(x+1,b)$ are given by the probability mass of the number of discovered nodes in round x.

[2] An approximate value of n can be calculated using node density.

Note that the states' probabilities are greatly dependent on the number of nodes n and the number of slots s. In fact, the probability $b_s^n(x,y)$ of the state (x,y) given that there are n nodes and s slots per round can be given as

$$b_s^n(x,y) = \begin{cases} q_y(n,s) & \text{if } x = 1 \\ \sum_{k=0}^{n} [q_{(k)}(n-(y-k),s) & \\ \times b_s^n(x-1,y-k)] & \text{if } 1 < x \leqslant r \end{cases}. \quad (11)$$

Therefore, the pdf of the number of neighbors discovered after r rounds is given by calculating the probabilities $b_s^n(r,0), b_s^n(r,1), \ldots, b_s^n(r,n)$.

4.3. Model validation

To validate the accuracy of our analytical models, we have implemented the slot selection mechanism in MATLAB and compared its result with the result obtained from the models presented earlier. Simulation results are averaged over 10,000 runs with different seeds. Fig. 10 shows a comparison of the cumulative distribution function of the number of discovered nodes out of 10 nodes using different number of slots. As shown in the figure, the model matches perfectly with the simulation results. Fig. 11 shows the cumulative distribution function of the number of discovered nodes using 4 slots in different number of rounds with similar accuracy.

4.4. Parameter optimization

The aforementioned Hello-Reply model can be used to optimize the parameters of the neighbor discovery mechanism. We are interested in finding the optimal system configuration that minimizes the discovery time such that probability that a node misses any of its n neighbors is less than a certain threshold ϵ. The expected number of neighbor nodes can be estimated using the estimated node density in combination with the estimated area coverage for each of the K sectors. Alternatively, an estimation algorithm [25,26] could be run prior to the proposed neighbor discovery algorithm which might add more time to the discovery process. Each round r takes $t_r + t_s{*}s$ seconds to finish where t_r is the time to send the Hello packet and t_s is the slot duration. Then, the discovery time can be calculated as

$$t_d = r * (t_r + t_s * s). \quad (12)$$

We can represent this optimization problem as the following linear program

$$\begin{array}{ll} \text{Minimize} & r * (t_r + t_s * s) \\ \text{Subject to} & P_{Miss} \leqslant \epsilon \end{array}, \quad (13)$$

where P_{Miss} is the probability of missing at least one node and can be computed using Eq. (11) as follows:

$$\begin{aligned} P_{Miss} &= 1 - P[n \text{ nodes are discovered in } r \text{ rounds}] \\ &= 1 - b_s^n(r,n). \end{aligned} \quad (14)$$

To solve this optimization problem, we propose a heuristic approach to find the optimal number of slots and rounds. The approach sequentially iterates through all slot values starting from 2 slots to *maxSlots* slots. Then, for each number of slots, the approach finds the minimum number of rounds that satisfies the condition $P_{Miss} \leqslant \epsilon$ and minimizes the discovery time. The algorithm **FindOptimalConfiguration(**n**,** ϵ**,** *maxSlots***,** t_r**,** t_s**)** describes our approach.

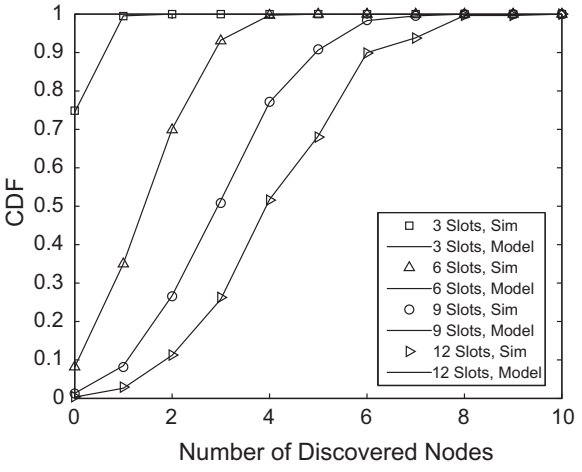


Fig. 10. CDF of the number of discovered nodes with 10 nodes and different number of slots in one round.

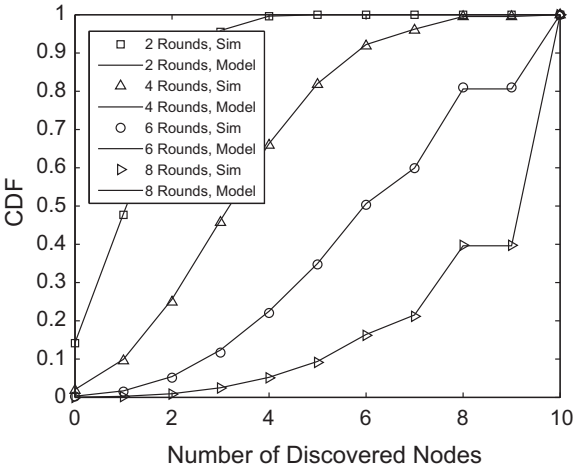


Fig. 11. CDF of the number of discovered nodes with 10 nodes, 4 slots and different number of rounds.

5. Simulation evaluation

5.1. Performance comparison

To verify the concepts of the proposed SAND protocol, we have implemented SAND in Qualnet simulator [27] using the directional antenna module emulating a four sectored antennas pattern shown in Fig. 21d. For a comparative study, we have implemented the asynchronous direct discovery mechanism proposed in [13], abbreviated throughput this section as Directional Neighbor Discovery DND, which does not depend either on omnidirectional

antennas nor on time synchronization. With SAND protocol, the neighbor discovery stops once the neighbor discovery token is received by the collecting node. The Hello-Reply parameters (i.e. N_{rounds} and N_{slots}) are optimized using the algorithm *FindOptimalConfiguration* where the number of nodes n are increased from 5 to 40 nodes and $\epsilon = 0.95$. To run *FindOptimalConfiguration*, the expected number of neighbors is estimated based on the antenna coverage and the density of the network. Nodes are distributed randomly on a 100 m $\times$ 100 m field with transmitting power of 10 dBm making a multihop network of two hops.

With DND the time is slotted. At the beginning of each time slot, each node transmits a discovery beacon in a random sector (direction) with probability p_t and listens for other beacon transmissions with probability $1 - p_t$. Since DND is a distributed protocol, we have used two conditions to stop the operation of DND. The first condition is to stop DND as soon as all nodes discover all their neighbors over any S2S links. With the second condition, DND stops as soon as the best Sector-to-Sector Links between all neighbors are discovered. Throughout the simulation studies, we call the second implementation of DND as Advanced DND (ADND). DND performance greatly depends on the Hello packets' sending rate which is optimized according to Eq. (11) in [13].

Figs. 12, 13 show the performance comparison between the three discovery protocols. Fig. 12 shows the number of control packets needed to perform the neighbor discovery. Note that for SAND, this number includes the control packets for the exchanging the tokens and neighborhood information. As shown in the figure, ADND requires higher number of control packets compared to DND and SAND. DND requires comparable amount of overhead as SAND but its requirements highly varies. Fig. 13 shows the average Received Signal Strength RSSI per link per node as the network density increases. The results for ADND has been omitted since it discovers, like SAND, the best S2S links between neighbor nodes. On the other hand, DND protocol stops once it discovers all neighbors without considering the RSSI of the links.

To further demonstrate the benefits of careful S2S link selection through SAND, in Figs. 14–16 we show the average link quality improvement. In these figures, 20 nodes

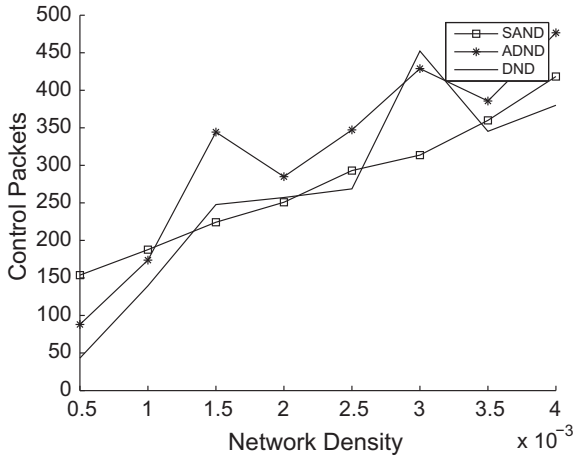


Fig. 12. Per node control packets.

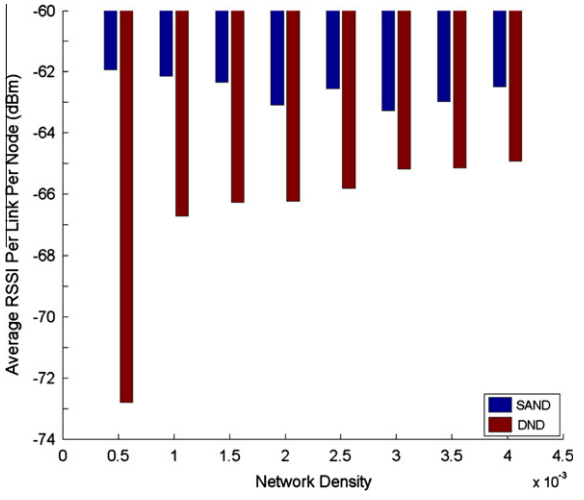


Fig. 13. Per node average link RSSI.

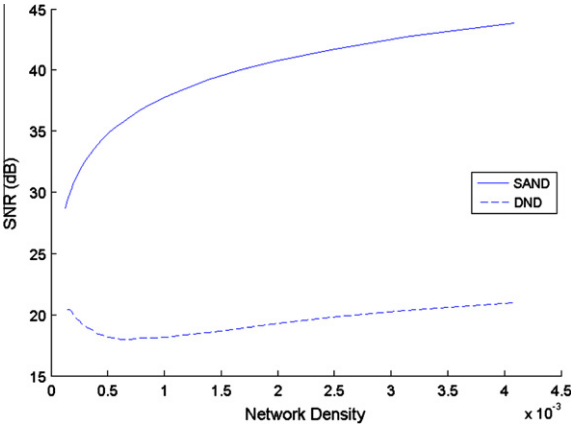


Fig. 14. Improved SNR of SAND vs. DND.

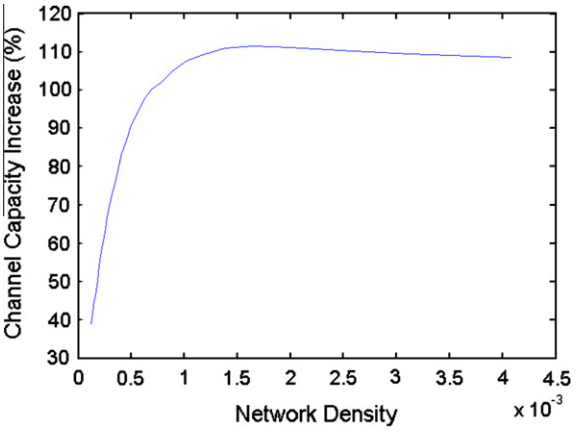


Fig. 15. Channel capacity improvement of SAND vs. DND.

were simulated with varying network simulation areas to model the changing network density. In each simulation set, the SAND protocol gathers channel quality information each node pair and for all S2S links in which communication is possible. The SAND protocol then utilizes the best link, in terms of SNR, to improve network performance. Due to the randomness of the DND protocol, neighboring nodes will discover each other on the first viable S2S link in which communication is established. Therefore, the expected SNR of any communication pair discovered through DND is the average SNR of the S2S links that can support communication. In Fig. 14 we see that in sparse networks with nodes spread far apart, the benefit of SAND is limited. As the nodes are spread far apart, the non-ideal S2S communication links are weakened to the point where they cannot support communication. Therefore, as the number of viable S2S communication links is reduced, the probability of DND finding the optimal S2S link is increased. However, as the network density is increased, more non-optimal S2S links become discoverable and we see that by carefully selecting the optimal S2S link, we see significant improvements in terms of SNR.

FindOptimalConfiguration(n, ϵ, *maxSlots*, t_r, t_s)**:** For a given n, find the minimum discovery time $r*(t_r + t_s*s)$ such that $P_{Miss} \leqslant \epsilon$

Input: n, ϵ, *maxSlots*, t_r, t_s
Output: $s_{optimal}$, $r_{optimal}$
begin procedure
1. *DiscoveryTime* = ∞
2. **for** $s = 2$ **to** *maxSlots* **do** /* Slots For Loop*/
3. *notFound* = *TRUE*
4. $r = 1$ /* Start with one round*/
5. **while** *notFound* **do** /* Rounds While Loop*/
6. $P_{Miss} = 1 - b_s^n(r, n)$ /* using Eq. (11) */
7. *newDiscoveryTime* = $r * (t_r + t_s * s)$
8. **if** ($P_{Miss} \leqslant \epsilon$) **and** (*newDiscoveryTime* $\leqslant$DiscoveryTime)
9. *DiscoveryTime* = *newDiscoveryTime*
10. $s_{optimal} = s$
11. $r_{optimal} = r$
12. *notFound* = *FALSE*
13. **else**
14. $r = r + 1$ /* Increase the number of rounds*/
15. **end if**
16. **end while** /* Rounds While Loop*/
17. **end for** /* Slots For Loop*/
end procedure

To better demonstrate this improvement, we show in Fig. 15 the improvement in peak channel capacity as modeled by the Shannon–Hartly theorem. Improvements of as much as 110% are possible in these scenarios by utilizing the SAND protocol. To get a more realistic view of how the improved SNR through SAND neighbor discovery can improve performance, we equate the improved SNR to channel data rate by utilizing well-known BER curves [28] for some reasonable modulation schemes. Assuming

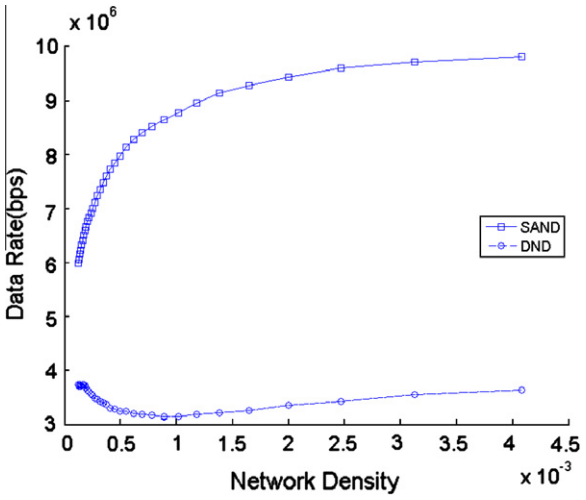


Fig. 16. Improved data rate of SAND vs. DND.

a base symbol rate of 1 mega-symbol per second and a minimum BER threshold of 10^{-5}, the following modulations were utilized:

Modulation	Data rate (Mbps)	Min SNR (dB)
BPSK	1	7
QPSK	2	12
16-QAM	4	20
64-QAM	6	26
256-QAM	8	31
1024-QAM	10	37

In Fig. 16 we see that, through SAND neighbor discovery, the optimal S2S links for these tologies support the up to 1024-QAM modulation at a data rate of 10 Mbps whereas the DND expected S2S links support data rates of only 2–4 Mbps.

In our simulations, the discovery time for single node, with a network density of 3×10^{-3} nodes per square meter, took approximately 1 s. As SAND is a serialized protocol, neighbor discovery time for the entire network increases linearly with the number of nodes in the network. SAND can be used in small-scale static mesh or ad-hoc networks. For large-scale WSNs, SAND can be run inside multiple clusters in parallel if the WSN is pre-partitioned into clusters. However, the advantages of SAND over DND and ADND are clear: with comparable control overhead, SAND discovers optimal S2S link pairs for neighboring nodes and collects this information at a centralized location. Moreover, DND and ADND's practical implementations still suffer from uncertainty of discovery durations since there is no centralized oversight.

5.2. Parameter evaluation

To test how the various control parameters (antenna beamwidth, N_{slots}, N_{round}) affects the performance of the SAND protocol, a simulation was developed using the

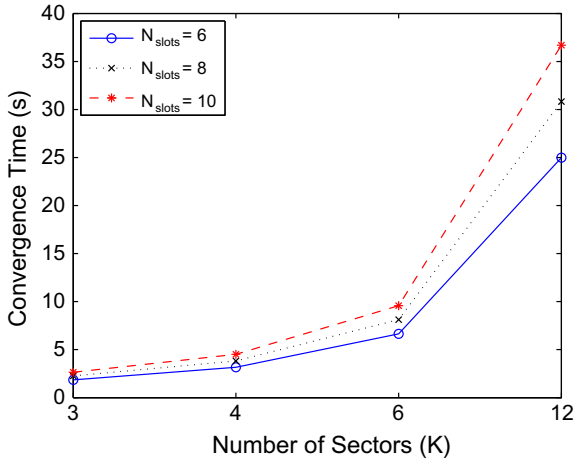
OPNET [29] simulator. Four sectorized antenna models were used, 120°, 90°, 60°, and 30°. For each antenna model, the number of sectors was selected to fully cover the 360° azimuth around a node. For each simulation, the following parameters were used, except where otherwide noted:

•Dimension	100 × 100 m
•Number of Nodes	20
•Transmission Power	1 mW
•Data Rate	1 Mbps
•Frequency	2.4 GHz
•Bandwidth	5 MHz
•t_{switch}	0.5 ms
•t_{HoneIn}	0.5 ms
•N_{HoneIn}	$K+1$
•$t_{replySlot}$	0.5 ms

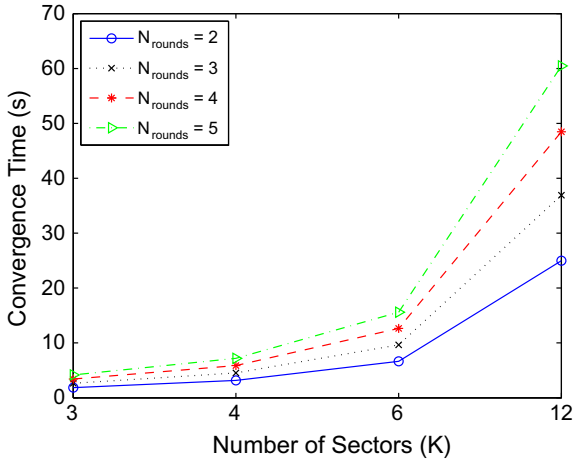
In Fig. 17, the convergence time of the SAND protocol is shown for increasing values of K. This time represents the duration from when the sink node begins neighbor discovery and when the last discovered node finishes reporting its local neighborhood information to the sink node. With increasing K the convergence time increases as well as the number of control packets generated in all phases in increased, as shown in Fig. 18. However, based on Figs. 17 and 18 we see that increasing N_{rounds} has a more significant impact on both the convergence time and overhead than N_{slots}.

In Fig. 19 the percentage of discovered links is shown. Again, we see that increasing N_{rounds} has a more significant impact on the discovery percentage than N_{slots}. Furthermore, for larger values of K, the discovery ratio is improved significantly. With larger values of K, the beamwidth of the individual antenna sectors is reduced. For a given network density, the narrow beamwidth antennas should result in less neighbors per sector. Therefore, with increasing K, there are less neighbors contending for the Hello-Reply slots, increasing the portion of nodes which select unique Hello-Reply slots.

Based on the results in Figs. 17–19 we have a better understanding of how to optimize parameter selection

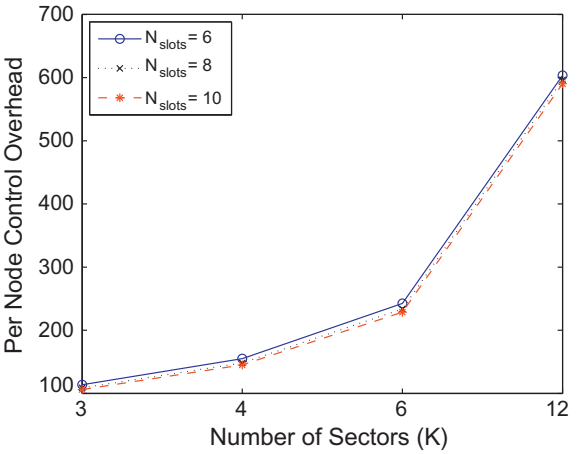


(a) $N_{rounds} = 2$

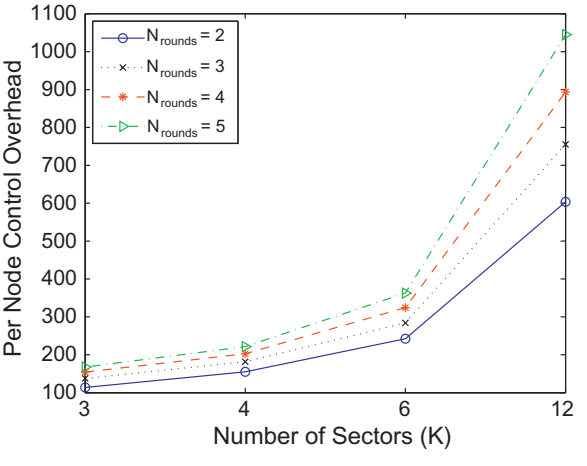


(b) $N_{slots} = 6$

Fig. 17. SAND convergence time vs. number of sectors.



(a) $N_{rounds} = 2$



(b) $N_{slots} = 6$

Fig. 18. SAND per-node control overhead vs. number of sectors.

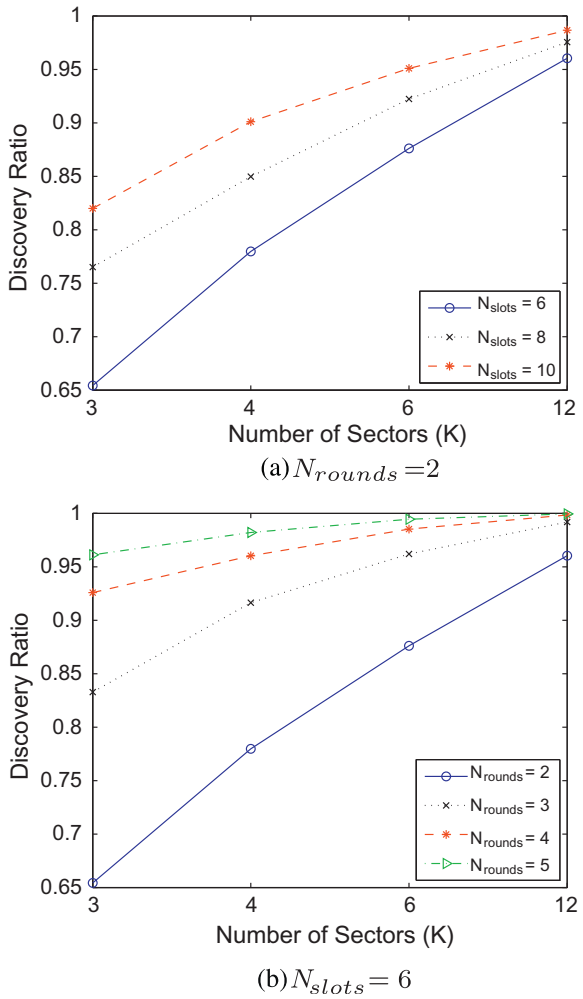


Fig. 19. Neighbor discovery ratio vs. number of sectors.

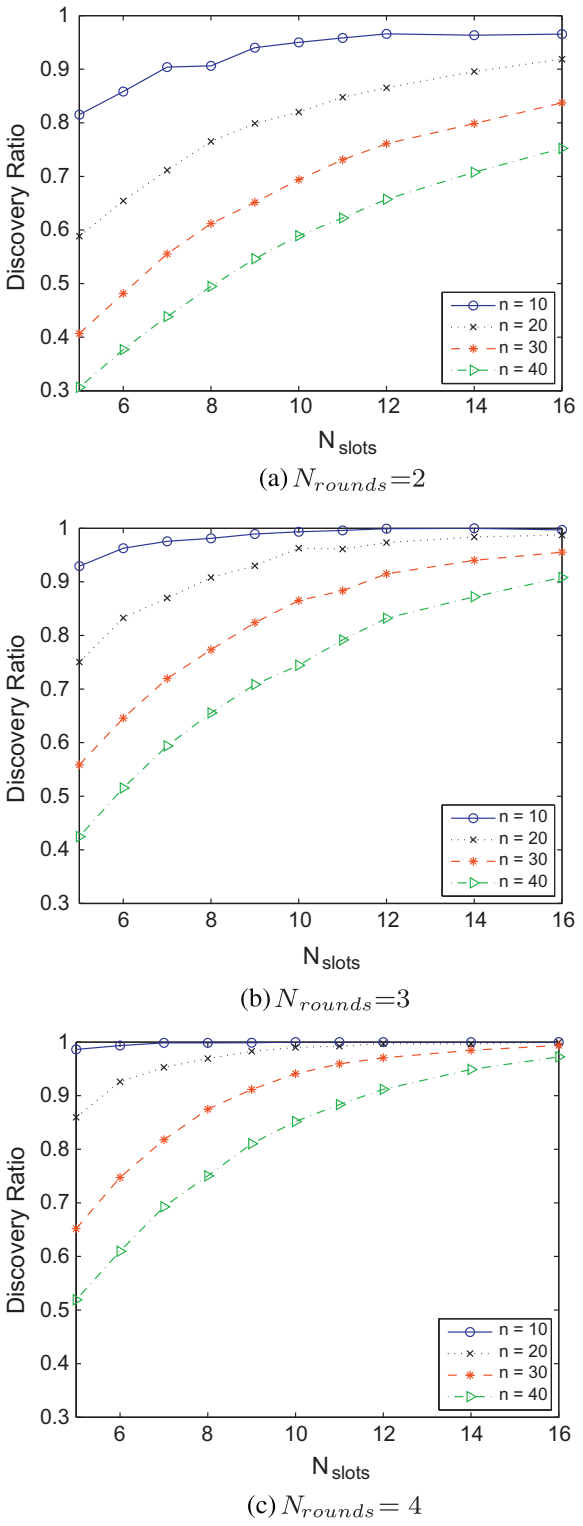
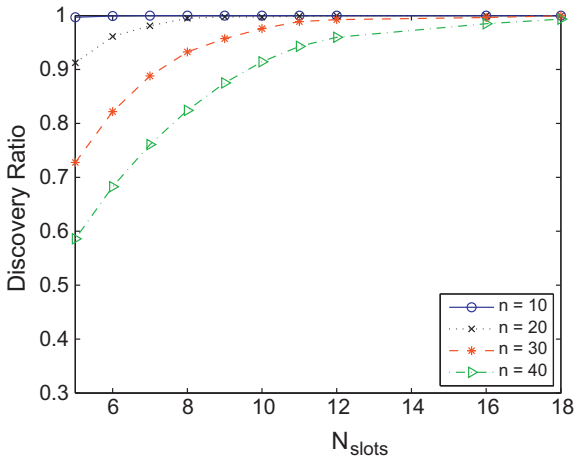


Fig. 20. Neighbor discovery ratio vs. N_{slots}.

for SAND. To reduce the amount of overhead (both in terms of control packets and convergence time) N_{rounds} should be minimized by increasing N_{slots} to reach a desired discovery ratio. For example, given a network of nodes utilizing a 120° sectorized antenna (K = 3) the discovery ratio for four different network densities is shown in Fig. 20. For these results, the network density was varied by simulating different amounts of nodes (n = 10, 20, 30, 40). Given these results and a desired discovery ratio of 90%, assuming that the maximum number of Hello-Reply slots is 16, the control parameters for the given results would be N_{rounds} = 3 and N_{slots} = 12. These parameters would yield the desired discovery ratio while minimizing N_{rounds}, and thus, minimizing the control overhead and convergence time of SAND. If the desired discovery ratio were increased to 98% for the same network, an additional Hello-Reply round would be required (N_{rounds} = 4 and N_{slots} = 14).

6. Testbed implementation

To validate SAND, we have implemented our protocol on a custom hardware platform developed by ETRI shown in Fig. 21c. All nodes use Texas Instruments (TI) MSP430 series processor for computation, TI CC2420 transceiver for communications, and a six-sectored antenna with overlapping 120° azimuth. Figs. 21a and b show the top and



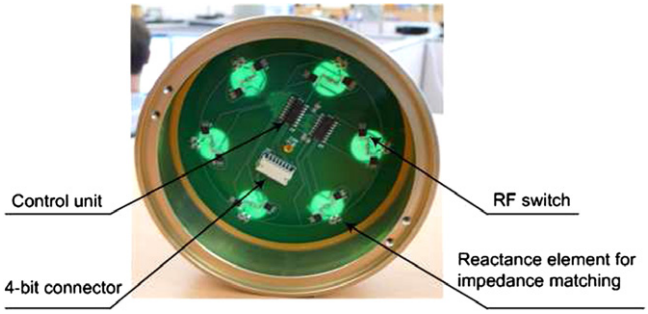
(d) N_{rounds}=5

Fig. 20 (*continued*)

bottom view of the sectored antenna, respectively. In the experiments, we only use three sectors to cover the entire azimuth. SAND was programmed using a proprietary operating system Nano-Q-Plus developed by ETRI. The testbed implementation of SAND was configured with the following parameters: transmission power is 0 dBm (1 mW), t_{switch} is 10 ms, t_{HoneIn} is 5 ms, N_{slots} is 5, and N_{rounds} is 5.

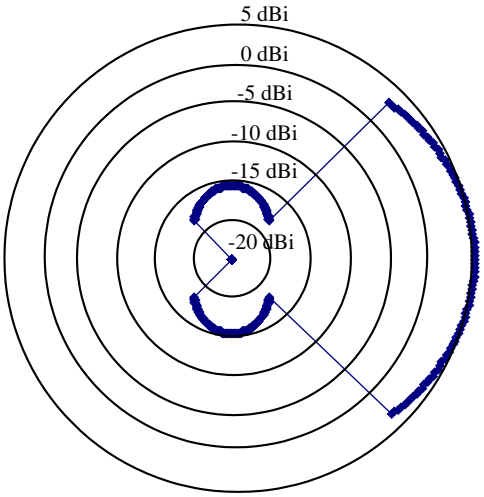
SAND was tested in an outdoor environment using a topology where five nodes (Nodes 1–5) are placed at random distances between 50 m and 85 m from a sink node (Node 0). Nodes 1–5 are oriented such that their sectors with Sector ID 1 point towards the sink node. Neighbor information was collected at each node and reported to the sink node. Fig. 22 shows the neighbor relations discovered at Node 0 for its neighbor Node 4 which were 52 m apart. The first two columns represent the sectors of Node 0 and 4, respectively. RSSI refers to the signal strength observed. Fig. 23 shows the S2S links that were discovered for all neighbor relations not including the sink node (node 0). Due to the spatial separation of the nodes, not all nodes are

(a) Sectored Antenna Top View



(b) Sectored Antenna Bottom View

(c) Hardware Platform With Sectored Antenna



(d) Antenna Gain Pattern for Simulations

Fig. 21. Testbed/simulation antenna implementations.

Node 0 Sector	Node 4 Sector	RSSI
2	1	-69
3	1	-74
2	2	-77
2	3	-77
1	1	-78
3	2	-81
3	3	-81
1	2	-84
1	3	-87

Fig. 22. All S2S links between Nodes 0 and 4.

(Node,Sec)	(Node,Sec)	RSSI
(1,3)	(4,1)	-82
(2,1)	(4,3)	-84
(2,1)	(3,1)	-84
(2,1)	(1,1)	-85
(4,2)	(5,3)	-83

Fig. 23. Best S2S links between remaining node pairs.

neighbors, as shown by the relatively brief list of neighbor relations.

Due to the proximity of Nodes 0 and 4, we can immediately see the need for careful selection of sectors for communication. Though there are multiple sector pairs that can support some level of communication, by testing and selecting the best S2S link between the neighboring nodes, as is done with SAND, we can see in this example an increase of 18 dB in RSSI. These additional neighbor relations, which are discovered through SAND, can be useful to TDMA scheduling schemes such as [24] to avoid unexpected interference due to improper sector reuse. For example, without this information, the scheduler could assign communication on Sector 2 of Node 4 and Sector 1 of Node 0 simultaneously, oblivious to the fact that, due to the presence of side-lobes in the sector antennas, these sectors can interfere.

7. Practical considerations

While implementing SAND on hardware, one must consider many practical issues that go beyond the theoretical development of the protocol. Some of the more significant considerations include (i) imperfect synchronization, (ii) hardware and operating system related delays, memory and bandwidth limitations, (iii) realistic sector antenna patterns that include more than just the main antenna lobe, and finally (iv) token recovery due to lost packets during neighbor discovery.

Imperfect synchronization is not only a problem between nodes, but within the operating system of the node itself. For example, the hardware this research was implemented on utilizes a 32.768 KHz crystal to synchronize its internal timers. The internal interrupts therefore have a granularity of approximately 32.768 μs. When utilizing hardware timers, the protocol can be fairly accurate in its timing, assuming that the processor is not busy when the hardware timer expires, though the number of available hardware interrupt timers is limited.

Memory and bandwidth limitations can have a significant impact on the performance of a neighbor discovery protocol, especially when the protocol, like SAND, is designed to collect link quality information on all discovered sector pairs. In this case, there can be a maximum of K^2 neighbor table entries for each neighbor. Assuming there is sufficient memory in each node for such neighbor tables, as the information is transferred to a central location for processing, the burden, in terms of memory and bandwidth requirements, will increase dramatically in multihop topologies. If only best S2S links are needed, it is possible to filter the neighbor tables locally, eliminating entries that correspond to weak links and thus reducing the bandwidth and memory burden for all nodes in the network significantly.

One must keep in mind when developing any neighbor discovery protocol with sector antennas that realistic sector antenna patterns have side lobes. For a neighbor discovery protocol, this translates to non-optimal sector combinations being discovered for neighboring nodes. SAND thoroughly tests all sector combinations for each neighbor to ensure that the optimal sector is discovered. While this may lead to increased burden on the network due to enlarged neighbor table, as described above, in the end this leads to a more reliable network as we ensure that all neighbors communicate through the sectors with the most favorable link conditions.

Finally, in token based protocols, such as SAND, one must always account for the possibility of a lost token. Unreliable wireless links or failed wireless nodes can easily cause breaks in the temporary depth-first routing tree. Such losses can be dealt with through timeouts. When passing the neighbor discovery token to a neighbor node, a timer can be set that expires after the expected token return time. After the timer expires, the token holder can resume the token-passing scheme, ignoring the bad link.

8. Conclusions

In this paper, we presented a fully directional neighbor discovery protocol called Sectored-Antenna Neighbor Discovery (SAND) protocol. Unlike many proposed directional neighbor discovery protocols, SAND depends neither on omnidirectional antennas to bootstrap the discovery process nor on time synchronization. In addition, SAND performs neighbor discovery in a serialized fashion allowing individual nodes to discover all potential neighbors within a predetermined time. Moreover, SAND systematically discovers all S2S links between neighboring nodes and gathers the neighborhood information in a centralized location, if needed, to be used by centralized networking

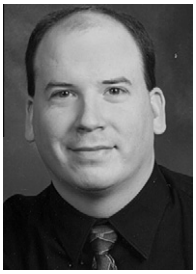
protocols. Effectiveness of SAND has been assessed via simulation studies and a real hardware implementation.

Acknowledgments

This research was supported by the Dual Use Technology Program, South Korea (06-II-LC-01, Surveillance and Reconnaissance Sensor Network Development).

References

[1] Ko Y, Shankarkumar V, Vaidya NH. Medium access control protocols using directional antennas in adhoc networks. IEEE INFOCOM 2000;2000:13–21.
[2] Ramanathan R. On the performance of ad hoc networks with beamforming antennas. In: MobiHoc '01, New York, NY, USA. ACM; 2001. p. 95–105.
[3] Choudhury RR, Yang X, Ramanathan R, Vaidya NH. Using directional antennas for medium access control in ad hoc networks. In: MobiCom '02, New York, NY, USA. ACM; 2002. p. 59–70.
[4] Yi S, Pei Y, Kalyanaraman S. On the capacity improvement of ad hoc wireless networks using directional antennas. In: MobiHoc '03, New York, NY, USA. ACM, 2003. p. 108–16.
[5] Wang Y, Garcia-Luna-Aceves J. Spatial reuse and collision avoidance in ad hoc networks with directional antennas. In: GLOBECOM '02, vol. 1; November 2002. p. 112–6.
[6] Sekido M, Takata M, Bandai M, Watanabe T. A directional hidden terminal problem in ad hoc network mac protocols with smart antennas and its solutions. In: GLOBECOM '05, vol. 5; December 2005. p. 5, p. 2583
[7] Gossain H, Cordeiro C, Cavalcanti D, Agrawal D. The deafness problems and solutions in wireless ad hoc networks using directional antennas. In: GLOBECOM '04; November 3–December 2004. p. 108–13
[8] Ramanathan R, Redi J, Santivanez C, Wiggins D, Polit S. Ad hoc networking with directional antennas: a complete system solution. In: IEEE journal on selected areas in communications, vol. 23; March 2005. p. 496–506
[9] Santosa R, Lee B, Yeo C, Lim T. Distributed neighbor discovery in ad hoc networks using directional antennas. In: IEEE CIT '06, vol. 1; 2006. p. 97.
[10] S. Zhang, A. Datta, A directional antenna based MAC protocol for wireless sensor networks, ICCSA2005 (2005) 686–695.
[11] Jakllari G, Lup W, Kirshnamurthy S. An integrated neighbor discovery and MAC protocol for ad hoc networks using directional antennas. In: IEEE transaction on wireless communication, vol. 6; March 2007. p. 1114–24.
[12] Z. Zhang, B. Li, Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons, IEEE Trans Wireless Commun 7 (2008) 1540–1549.
[13] Vasudevan S, Kurose J, Towsley D. On neighbor discovery in wireless networks with directional antennas. In: IEEE INFOCOM 2005, vol. 4; March 2005. p. 2502–12.
[14] Kumar U, Gupta H, Das S. A topology control approach to using directional antennas in wireless mesh networks. In: ICC 2006, vol. 9; June 2006. p. 4083–88
[15] T. Dimitriou, A. Kalis, Efficient delivery of information in sensor networks using smart antennas, Springer, Berlin, Heidelberg, 2000.
[16] Ko Y-B, Shankarkumar V, Vaidya N. Medium access control protocols using directional antennas in ad hoc networks. In: INFOCOM '00, vol. 1; 2000. p. 13–21
[17] Korakis T, Jakllari G, Tassiulas L. A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks. In: ACM MobiHoc '03; June 2003. p. 98–107.
[18] Nasipuri A, Ye S, You J, Hiromoto R. A MAC protocol for mobile ad hoc networks using directional antennas. In: IEEE WCNC 2000, vol. 3; 2000. p. 1214–9.
[19] Saha A, Johnson D. Routing improvement using directional antennas in mobile ad hoc networks. In: GLOBECOM '04, vol. 5; November–3 December 2004. p. 2902–8.
[20] Roy S, Saha D, Bandyopadhyay S, Ueda T, Tanaka S. A network-aware mac and routing protocol for effective load balancing in ad hoc wireless networks with directional antenna. In: MobiHoc '03, New York, NY, USA. ACM; 2003. p. 88–97.
[21] Namboodiri V, Gao L, Janaswamy R. Power efficient topology control for wireless networks with switched beam directional antennas. In: IEEE MASS '05; November 2005. p. 8, p. 596
[22] Huang Z, Shen C-C, Srisathapornphat C, Jaikaeo C. Topology control for ad hoc networks with directional antennas. In: IEEE ICCCN '02; October 2002. p. 16–21
[23] Pei G, Kim A, Nast J, Norris D, Norris P. A neighbor discovery protocol for directional antenna networks. In: IEEE military communications conference MILCOM, vol. 1; October 2005. p. 487–92
[24] Felemban E, Vural S, Murawski R, Ekici E, Lee K, Moon Y, Park S. Samac: a cross-layer communication protocol for sensor networks with sectored antennas. IEEE Trans Mobile Comput 2010;9(August): 1072 –88.
[25] Kodialam M, Nandagopal T. Fast and reliable estimation schemes in rfid systems. In: Proceedings of the 12th annual international conference on mobile computing and networking, MobiCom '06, New York, NY, USA. ACM; 2006. p. 322–33.
[26] A.G. Greenberg, P. Flajolet, R.E. Ladner, Estimating the multiplicities of conflicts to speed their resolution in multiple access channels, J ACM 34 (1987) 289–325.
[27] Scalable network technologies. <http://www.scalable-networks.com/>.
[28] Howald RL. Qam bulks up once again: modulation to the power of ten. White Paper, Motorolla Broadband Communications.
[29] OPNET technologies. <http://www.opnet.com/>.

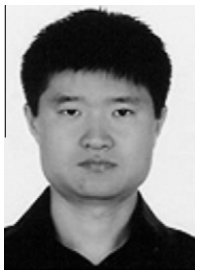


Robert Murawski received his BS degree in Electrical Engineering in 2003, and his MS degree in Electrical Engineering in 2004. He then worked in the area of wireless networking, developing wireless communication networks for the United States Military. In 2007 he returned to academia and is currently pursuing a PhD degree in Electrical Engineering from The Ohio State University.

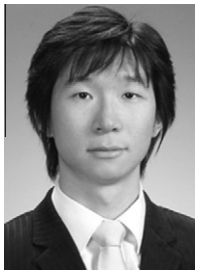
Emad Felemban (S'00-M'10) is an Assistant Professor at the Department of Computer Engineering, Umm al-Qura University (UQU), Makkah, Saudi Arabia. He received his master and PhD degrees from The Ohio State University in 2003 and 2009, respectively. His current research interests are in wireless sensor networks, performance analysis of wireless networks and QoS provisioning. He served as a TPC members for PIMRC 2010 and Globecom 2010.

Eylem Ekici (S'99-M'02) received the BS and MS degrees in Computer Engineering from Bogaziçi University, Istanbul, Turkey, in 1997 and 1998, respectively, and the PhD degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, in 2002. Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, The Ohio State University. His current research interests include wireless sensor networks, vehicular communication systems, cognitive radio networks, resource management, and analysis of network architectures and protocols. He is an Associate Editor of Computer Networks Journal (Elsevier) and ACM Mobile Computing and Communications Review.

Sangjoon Park received his BS, MS degrees in Electronics Engineering from the Kyung-Pook National University in 1988, and 1990 respectively. He received his PhD degree in Computer Science Department from North Carolina State University in 2006. He is currently Head of the Positioning Information Technology Research Team in Electronics and Telecommunications Research Institute. He was the Project Manager of the Surveillance and Reconnaissance Sensor Network Project which is supported by both Ministry of National Defense and Ministry of Knowledge Economy. His current research interests include wireless sensor network, next-gen embedded sensor network, multi-sensor data fusion, target tracking and positioning technology.



Seung-mok Yoo received his BS and MS degrees in Computer Engineering from Kyungpook National University, Daegu, Korea in 1994 and 1996, respectively. He received the PhD degree in Electrical and Computer Engineering from University of California, Irvine in 2007. He was a researcher at Agency for Defense Development, Korea from 1996 to 2001. He is currently a senior engineer at Electronics and Telecommunications Research Institute, Korea. His research interests include node architecture design, MAC and routing protocol design, distributed real time system design and implementation in wireless sensor networks and embedded systems.



Kangwoo Lee received the BEng and MS degrees in Computer Science and Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2005 and 2007, respectively. His master's research focused on remote code update for wireless sensor networks. He is currently working for the Electronics and Telecommunications Research Institute (ETRI) in Korea. His current research interests include MAC protocol in wireless sensor networks with a special emphasis on dependability of embedded systems.

Ju-Derk Park received his BS, MS degrees in Computer and Communication Engineering from Chungbuk National University, Korea, in 1997, 1999, respectively. Since 2000, he has worked on RFID and wireless sensor networks in ETRI. Now he is senior researcher of Intelligent sensor network research team, which is researching and developing the key technologies of wireless sensor network in Military technology area, especially.

Zeeshan Hameed Mir is an Engineering Staff Member at Electronics and Telecommunication Research Institute (ETRI). Prior to joining ETRI in 2009, he received his PhD degree in Information and Communication Engineering from Ajou University, South Korea. He received his B.S. degree in Computer Engineering from the Sir Syed University of Engineering and Technology (SSUET), Karachi, Pakistan in 1999 and M.S. degree in Computer Engineering from the National University of Sciences and Technology (NUST), Rawalpindi, Pakistan, in 2004. His interests include routing and MAC in multi-hop wireless ad hoc/sensor networks, cross-layer control protocol design and implementation and QoS provisioning.