
AFRL-RI-RS-TR-2019-065

# DESIGN OF NOVEL, CROSS-LAYER NEIGHBOR DISCOVERY SCHEMES FOR DIRECTIONAL MESH NETWORKS

SAN DIEGO STATE UNIVERSITY FOUNDATION

*MARCH 2019*

FINAL TECHNICAL REPORT

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED***

STINFO COPY

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-065 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

**/ S /**
GREGORY HADYNSKI
Work Unit Manager

**/ S /**
QING WU
Technical Advisor, Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| MARCH 2019 | FINAL TECHNICAL REPORT | APR 2017 – OCT 2018 |

**4. TITLE AND SUBTITLE**

DESIGN OF NOVEL, CROSS-LAYER NEIGHBOR DISCOVERY SCHEMES FOR DIRECTIONAL MESH NETWORKS

**5a. CONTRACT NUMBER**
N/A

**5b. GRANT NUMBER**
FA8750-17-1-0140

**5c. PROGRAM ELEMENT NUMBER**
62788F

**6. AUTHOR(S)**

Sunil Kumar

**5d. PROJECT NUMBER**
T2MA

**5e. TASK NUMBER**
SD

**5f. WORK UNIT NUMBER**
SU

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
San Diego State University Foundation
5250 Campanile Drive, MC 1901
San Diego, California 92182

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory/RITF
525 Brooks Road
Rome NY 13441-4505

**10. SPONSOR/MONITOR'S ACRONYM(S)**
AFRL/RI

**11. SPONSOR/MONITOR'S REPORT NUMBER**
AFRL-RI-RS-TR-2019-065

**12. DISTRIBUTION AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

This report discusses the design of three novel, cross-layer neighbor discovery schemes for directional wireless mesh networks, which have significantly lower protocol overhead and discovery latency. These schemes intelligently consider the collisions among the neighbor discovery messages of the neighboring nodes and use machine learning techniques. The performance of these schemes is evaluated for different network sizes, node densities, beamwidth and number of one-hop neighbors. Finally, implementation of directional neighbor discovery schemes in a hardware testbed is discussed.

**15. SUBJECT TERMS**

Wireless mesh networks, neighbor discovery, directional communication, cross-layer, USRP boards.

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | **GREGORY HADYNSKI** |
| U | U | U | UU | 54 | 19b. TELEPHONE NUMBER (*Include area code*) **N/A** |

# Table of Content

## List of Figures

# 1. SUMMARY

This project began on April 14, 2017 and ended on October 14, 2018. The main goal of this project was to design novel neighbor discovery protocols for nodes equipped with directional antennas in a wireless mesh network (WMN). Since the nodes in self-configuring networks, such as WMN, are deployed in an ad hoc manner, they need to discover their one-hop neighbors for data communication. Therefore, neighbor discovery is an important first step in the network setup. The neighborhood information is critical for network operations, including the topology management, routing, clustering and medium access control (MAC) layer operation. Note that the neighbor discovery is trivial when omni-directional transmission is used because a simple broadcast can reach all the nodes in a 1-hop neighborhood. However, the network consisting of purely directional-only antennas (without the support of omni-directional antennas) requires a significant neighbor discovery overhead and large latency because a node can communicate with its neighbors in only a narrow area at a time due to the limited antenna beamwidth.

In this project, we have designed three different novel directional neighbor discovery schemes which have significantly lower protocol overhead and discovery latency. These schemes intelligently consider the collisions among the neighbor discovery messages of the neighboring nodes and use the machine learning techniques. The performance of these schemes is evaluated for different network sizes, node densities, beamwidth and number of one-hop neighbors. We have also built a hardware testbed to test the performance of neighbor discovery schemes for WMN.

First, we have designed a novel *collision-aware* neighbor discovery scheme based on the two-way handshaking, which introduces a collision resolution mechanism that allows the neighbor discovery process to converge significantly faster, especially in the case of high node density. We also propose a *peer-assisted* version of our algorithm where nodes cooperate by sharing the information about their already discovered common neighbors to maximize the discovery rate.

Next, we have designed a *fully directional neighbor discovery scheme based on finite-state learning automata*. Every node learns about its neighborhood in each sector, and chooses the next sector by taking into account collisions and previously discovered neighbors. To the best of our knowledge, the learning automaton has so far not been applied to neighbor discovery with fully directional links.

Finally, we have designed a novel distributed Q-learning based directional neighbor discovery scheme, which gives the nodes the ability to learn about their respective neighborhood through past discovery attempts in each sector. We first map the neighbor discovery process to a model-free learning environment, followed by the use of a Q-learning algorithm to minimize the directional neighbor discovery latency. In every time slot, the node interacts with its neighborhood by executing an action and receives a reward. Based on the information collected, a node adjusts its operating mode to reduce the neighbor discovery latency.

It is important to verify the directional neighbor discovery protocols through a realistic hardware platform. We have used National Instrument (NI) USRP-RIO boards, equipped with two RF interfaces, which can connect two directional antennas. Thus we can emulate a two-beam antenna. We have used Matlab and Labview programming to configure multiple USRP-RIO nodes such that they can use different beams for neighbor discovery and data communication.

In the next chapter, we provide an introduction of directional neighbor discovery, including a review of existing literature. The directional neighbor discovery schemes in this project are described in Chapter 3, folowed by their performance analysis in Chapter 4. Conclusions are given in Chapter 5.

This project has achieved fruitful outcomes: one accepted IEEE journal paper and another manuscript under preparation, two published conference papers and third one under review. The detailed list is provided below:

**Journal Papers:**

1. B. El Khamlichi, D. H. N. Nguyen, J. El Abbadi, N.W. Rowe, and S. Kumar, *Learning Automaton Based Neighborhood Discovery for Wireless Networks Using Directional Antennas*, IEEE Wireless Comm. Letters, Accepted.
2. B. El Khamlichi, J. El Abbadi, and S. Kumar, *Distributed Q-Learning based Nieghbor Discovery in Directional Wireless Mesh Networks*, under preparation.

**Conference Papers:**

3. B. El Khamlichi, Duy H. N. Nguyen, J. El Abbadi, N. Rowe, and S. Kumar, *Collision-Aware Neighbor Discovery with Directional Antennas*, International Conf. Computing, Networking and Communications (ICNC 2018), Workshop on Computing, Networking and Communications, Maui, Hawaii, USA, March 5-8, 2018.
4. A. Shaha, Duy H. N. Nguyen, and S. Kumar, N. W Rowe, *Real Time Video Transceiver using SDR testbed with Directional Antennas*, IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017.
5. B. El Khamlichi, J. El Abbadi, and S. Kumar, Adaptive Directional Neighbor Discovery Scheme in Wireless Networks, submitted to IEEE WCNC.

# 2. INTRODUCTION

In order to address the challenges of a high-capacity backbone for future tactical edge networks, the IP- enabled, *directional*, mobile, mesh networks (WMN) are very promising. Figure 1 shows an example of a WMN architecture. The directional mesh networks provide the following benefits: high capacity/spectrum efficiency, scalability (new nodes bring additional channel capacity so they can scale to several hundred mutually non-interfering nodes without loss of per-node capacity), and inherent interference resilience [1]. These networks must be robust (i.e., they should have the properties of automatic node discovery, self-organizing and self-healing), with no single point of failure.

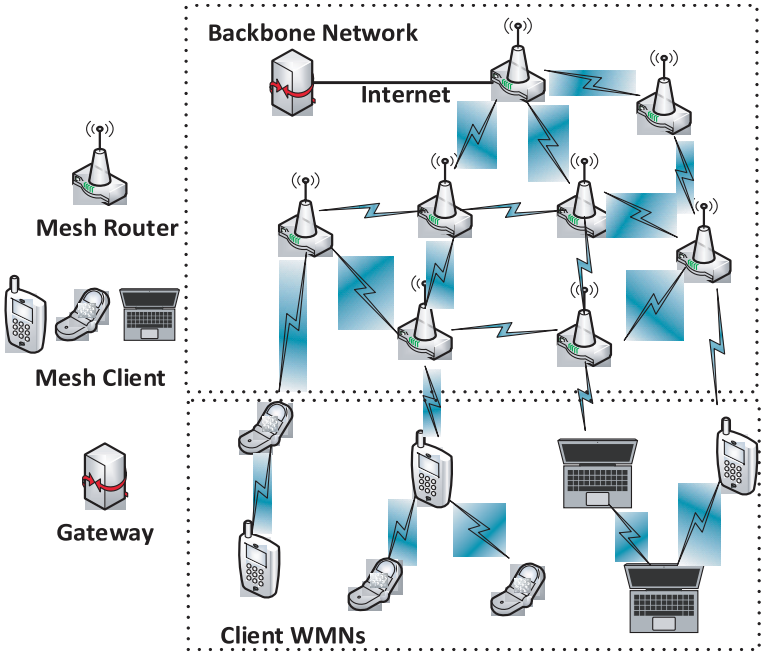


Figure 1. WMN architecture.

Increasing the antenna gain over an omni-directional antenna has multiple benefits, such as increasing the communication range, link data rate, and/or improving link reliability [1], [2], [3], [4]. To maximize these advantages and to enable spatial re-use, it is necessary to have both the transmitter and receiver use directional beams. This substantially reduces the probability that any two links will interfere with each other. However, the directional networks, in particular narrow-beam networks, challenge many basic assumptions for omni-directional wireless networks and require the design of novel cross-layer network protocols.

In self-configuring networks such as WMN, after nodes are deployed, they need to discover their one-hop neighbors for data communication. Neighbor discovery is an important first step in the initialization of WMN, as the accurate neighborhood information is critical for network operations, including topology management, routing, clustering and MAC operation [5], [6]. Note that the accurate neighbor discovery is trivial when omni-directional transmission is used because

a simple broadcast can reach all the nodes in a 1-hop neighborhood [7], [8]. However, the use of directional antennas complicates the neighbor discovery process due to the following reasons:

i. Limited beamwidth of directional antennas requires the neighbor discovery process to be repeated in different directions for covering the whole azimuth. This results in a significantly higher discovery overhead and latency.

ii. A pair of nearby directional nodes can discover each other only when their beams are oriented toward each other at the same time. Therefore, the neighboring nodes must know when and where to point their beams to discover each other. This is further complicated by node mobility.

In this project, we have developed and demonstrated the following three novel cross-layer neighbor discovery schemes for WMN nodes equipped with fully-directional antennas. We have also developed a hardware testbed to test the performance of neighbor discovery schemes for WMN.

Existing neighbor discovery schemes require high latency because all the sectors are explored randomly and collisions caused by multiple simultaneous communications are ignored. When more than one transmitting neighbor exists in the same reception beam, a collision happens and the node discovery packets are considered lost. Since the work in [6, 11] does not propose a management mechanism in case of collision, the neighbor discovery is generally simply reiterated. However, the collision effect should not be ignored, since collisions may happen frequently in many realistic scenarios, especially for the case of wide beam-width, high node density, or large transmission range.

To remedy this issue, we first design a novel, fully directional, *collision-aware* neighbor discovery scheme based on the two-way handshaking algorithm proposed in [6]. Our scheme introduces a collision resolution mechanism that will allow the neighbor discovery process to converge significantly faster, especially in the case of high node density. Second, we propose a *peer-assisted* version of our algorithm where nodes cooperate by sharing the information about their already discovered common neighbors to maximize the discovery rate and thereby reduce the discovery overhead and latency.

Next, we have designed a *fully directional neighbor discovery scheme based on finite-state learning automata*. In this scheme, every node learns about its neighborhood in each sector, and intelliegently selects the next sector by taking into account collisions and previously discovered neighbors. Our scheme outperforms the existing schemes in terms of neighbor discovery latency and convergence. To the best of our knowledge, the learning automaton has so far not been applied to neighbor discovery with fully directional links.

Finally, we have designed a novel distributed Q-learning based directional neighbor discovery scheme. The key idea behind this scheme is to give the nodes the ability to learn about their respective neighborhood through past discovery attempts in each sector. We first map the neighbor discovery process to a model-free learning environment, followed by the use of a Q-learning algorithm to minimize the directional neighbor discovery latency. In every time slot, the node interacts with its neighborhood by executing an action and receiving a reward. Based on the information collected, a node adjusts its operating mode to reduce the neighbor discovery delay. The contributions of this scheme can be summarized as: (a) The neighbor discovery process is modelled as a multi-agent model-free learning problem. Each node is regarded as an independent

agent gradually learning from its neighborhood and adapting its search strategy until reaching convergence. (b) The learning process is solved by means of the Q-learning algorithm, where a node adjusts its operating mode (i.e., to transmit or to receive) to reduce the overall discovery latency. (c) The Q-learning based scheme is compared to the random and deterministic discovery schemes and an extensive performance analysis shows that our scheme outperforms the existing schemes.

It is important to verify the directional neighbor discovery through a realistic hardware platform. We have used National Instrument (NI) USRP-RIO boards to construct a short-distance (<100m) multi-beam hardware platform [9]. The USRP-RIO board has two RF interfaces, which can connect two directional antennas. Thus we can emulate a two-beam antenna. We have used Matlab and Labview programming to configure multiple USRP-RIO nodes such that they can use different beams for directional neighbor discovery and data communication.

## 2.1. Related Work

Only a limited number of schemes are available in the literature to improve the neighbor discovery latency for nodes equipped with directional antennas. In some schemes (such as [10, 11, 12]), a secondary omnidirectional antenna is assumed to be present on each directional node to allow a faster discovery rate. However, omnidirectional antennas are known to be highly vulnerable to interference. Further, the combination of directional and omni-directional transmissions degrades the spatial reuse benefits, besides introducing range asymmetry problems.

The fully-directional neighbor discovery schemes that use directional control and data transmissions can be divided in four broad classes: integrated unassisted, non-integrated unassisted, peer assisted, and external assisted classes, as described below [1].

The i*ntegrated unassisted neighbor discovery* automatically discovers other nodes without prior knowledge of them using the same set of channel frequencies and directional antenna beams used for transmitting user and control data between nodes. The n*on-integrated unassisted neighbor discovery* automatically discovers other nodes without prior knowledge of them using a separate control channel for neighbor discovery and link establishment that covers all of the vicinity around a node within which a neighboring node might acquire or operate a link to that node. This approach requires access to an additional spectrum and adds implementation complexity but provides performance advantages compared to common frequency neighbor discovery. The *peer assisted neighbor discovery* uses the neighbor-provided nearest neighbor node position information for the neighbor discovery process. Here the nodes collaborate by sharing with each other the information about their already discovered neighbors. This reduces the neighbor discovery overhead. *External assisted neighbor discovery* takes advantage of the presence of any node position information available with a supervisory node (such as the cluster head) or over external interfaces. This reduces neighbor discovery overhead.

The fully-directional neighbor discovery schemes can also be classified based on the use of time-synchronization and asynchronous operation. Some neighbor discovery schemes assume that nodes are time synchronized to guarantee that all nodes switch their sectors synchronously. However, these schemes 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 very unlikely in WMN with dynamic topologies. In the asynchronous, fully-directional neighbor schemes, each node listens in a random direction for a

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

In [12], a randomized one-way handshaking mechanism was discussed for neighbor discovery in directional wireless networks. In each slot, a node probabilistically chooses to transmit or to receive in a randomly chosen sector (or direction). Due to the lack of reception confirmation, the discovered neighborhood for each node is asymmetric and implies a high latency. To deal with these issues, various two-way [13] and three-way handshaking [14] protocols were proposed. In [13], the authors proposed a random two-way handshaking mechanism and derived an analytical expression for the expected time to achieve a full neighbor discovery. The gossip-based version of the algorithm proposed in [15] enables a faster discovery through the cooperation between the nodes for exchanging neighbor information (assuming that GPS information is available). An improvement of 2-way handshaking mechanism was proposed in [13] through a selective feedback to reduce the collision probability during the feedback period. Our scheme proposed a collision resolution mechanism to allow a faster neighbor discovery in dense networks [16]. In [17], a more general and practical version of the neighbor discovery scheme was proposed for heterogeneous networks with different communication characteristics.

Another set of algorithms considered a deterministic approach, where each node operates in a predefined scanning sequence. In [12], a node operates in a predefined sequence based on its ID in deterministic scan based algorithm (SBA-D), whereas a random sequence is used in the random scan based algorithm (SBA-R). In[18], the authors carried out a detailed investigation of the neighbor discovery performance in a 60 GHz network equipped with directional antennas by taking into consideration the impact of the antenna pattern and link model. An improvement to the random scan algorithm was proposed in [19] by eliminating the unlikely sectors assuming that each node has knowledge about its position via GPS. An ALOHA-like algorithm was proposed in [15] by modeling the neighbor discovery process as a coupon collector problem. An asynchronous serialized centralized neighbor discovery scheme was proposed in [20]. In this scheme, the neighbor discovery is performed only by the node that holds the token, and the token is exchanged by nodes until they achieve the network-wide neighbor discovery. A full integration of neighbor discovery with MAC protocol was described in [3, 4, 21]. Here, the neighbor discovery mechanism is incorporated with a scheduling based medium sharing technique that allows for exclusive directional transmission and reception.

In the above mentioned algorithms, the neighbor discovery process still suffers from high protocol overhead and latency. *First*, most of these schemes ignore the effect of collision by assuming only one neighbor per beam or considering the colliding packets as lost. However, this assumption can lead to a drastic performance decline in the case of dense networks. *Second*, the nodes behave completely randomly in each discovery phase and do not consider the knowledge that can be gathered during past discovery attempts. Therefore, the random-search based approaches can be very time consuming. Our another scheme [22] has proposed a learning automaton based directional neighbor discovery (DND) protocol that is able to adapt the probability of exploring each sector based on the collision and successful discovery experienced in each sector. To the best of our knowledge, no other scheme has applied the reinforcement learning to the directional neighbor discovery problem so far.

# 3. METHODS, ASSUMPTIONS, AND PROCEDURES

## 3.1. Collision-Aware Directional Neighbor Discovery Scheme

### 3.1.1. System Model and Assumptions

We assume that each node is equipped with a steerable directional antenna with beamwidth $\beta = \frac{2*\pi}{K}$ and $(0 < \beta < 2\pi)$. To model the directional gain of antenna, we use the ideal keyhole model, which considers only the main lobe, i.e, the direction of maximum power for transmission or reception, while the side and back lobes are ignored.

We assume that a pair of nodes can communicate in the directional mode if their beams points toward each other. In order to formalize this condition along with the beam model, we define $\varphi$ and $\varphi'$ as the respective transmit and receive directions of the two nodes. We verify the previous condition through the following formula:

$$\varphi = (\varphi' + \pi) \mod 2\pi \quad (1)$$

Moreover, we consider that the nodes possess the following features:

- Each node is identified by a unique ID.
- All nodes are perfectly synchronized.
- The nodes are half duplex. At each slot, a node selects one direction for either transmission or reception.

### 3.1.2. Algorithm Description

#### 3.1.2.1. Transmission Strategy

The node discovery time is divided into slots and nodes are perfectly synchronized. Each time slot is further divided into two subslots, where the duration of a subslot is equal to the time required to transmit a discovery packet. At the beginning of each time slot, a node decides to transmit with a probability $P_t$ or to listen with probability $(1 - P_{t)}$ in a randomly chosen beam direction. If a node is in the transmitting mode, it will send an advertisement message with its ID in a given direction during the first sub-slot. In the second sub-slot, the node will listen in the same direction. On the other hand, when a node is in the listening mode, the first sub-slot is dedicated to receiving the potential neighbor discovery messages in a chosen direction. If a successful reception takes place and the node is not already discovered, it will send a reply back in the same direction in the second sub-slot. Otherwise, the node will remain silent.

#### 3.1.2.2. Collision Detection

A collision occurs when a node simultaneously receives two or more packets from its neighbors in a given beam. In this case, the node is not able to decrypt the information, and the packet is considered lost. As shown in Figure 2, a collision will take place in the first sub-slot if

the node is able to intercept more than one connection. For simplicity, we do not consider other packet loss factors. Each node can differentiate between an empty slot and a slot with a collision through a simple energy detector. While existing schemes do not consider the collisions occurring during the neighbor discovery process, our scheme exploits the collision information to improve the overall neighbor discovery performance as discussed below.

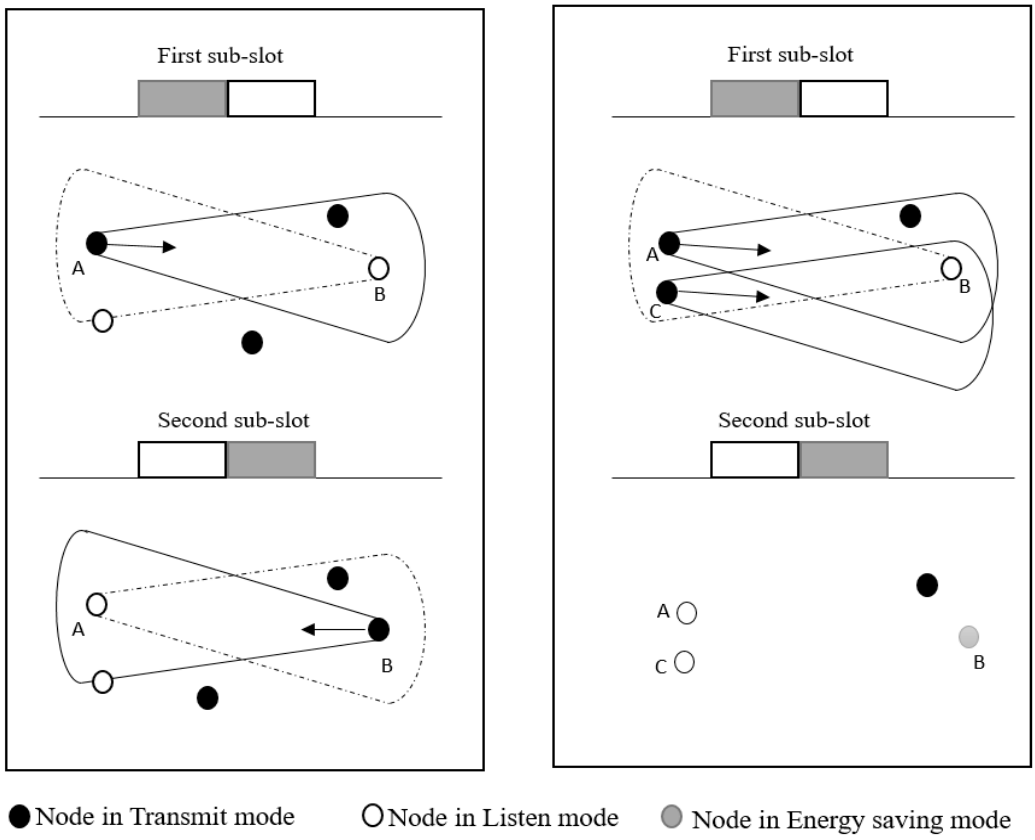


Figure 2. Successful and collision example (Adapted from Figure 7 in [23]).

### 3.1.2.3. Collision Resolution Mechanism

To illustrate how our algorithm handles collisions, we consider four nodes in a 1-hop neighborhood: A, B, C and D. As illustrated in Figure 3, during the first sub-slot, nodes A, C and D are in transmit mode while node B is listening. In this case, node B will receive multiple discovery packets which causes their collision. In the second subslot, node B uses the same beam to transmit a collision acknowledgment in its beam. The four nodes then enter in a collision resolution mode as explained below:

- **Node B :** The node B will remain in a listening mode until it successfully receives at least two neighbor discovery packets. We denote this mode as the "*collision-resolving listening mode*."
- **Node A, C, D:** The nodes A, C and D will plan the re-transmission in the same direction during the next slots. The re-transmission slot is chosen randomly within a contention window occupying $CW$ slots. During the other slots, the nodes can either transmit or listen in other directions. However, the "re-transmitting node" (say node A) cannot schedule any new re-

transmission towards node B until it detects a successful discovery acknowledgment in this beam. We denote this mode as the "*Collision-resolving re-transmitting mode*." If there are more than one node choosing the same slot to retransmit, a collision will occur again. Consequently, both nodes will plan another re-transmission phase in the next $CW$ slots and node B will remain in the listen mode.

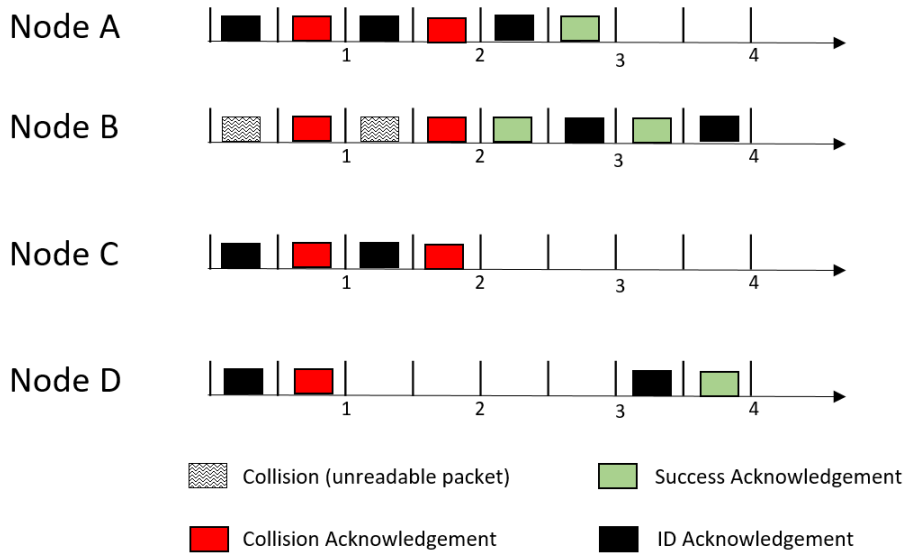


Figure 3. Illustration of the proposed collision resolution mechanism.

### 3.1.3. Analytic Formulation

#### 3.1.3.1. Transmission Probability, $P_t$

In the beginning of each time slot, every node in a 1-hop neighborhood randomly chooses to transmit or to listen with probability $P_t$ and $1 - P_t$, respectively. Intuitively, the performance of the node discovery process strongly depends on the choice of transmission probability. A high transmission probability will result in more collisions, and thus a slow discovery process. On the other hand, a low transmission probability will result in an underutilization of the channel and a slow discovery rate. In most of the previous probabilistic approaches, the optimal probability a node uses to achieve the best discovery rate is related to the number of neighbors. As proved in [24], the optimal probability is given by,

$$P_t = \frac{4*\Pi^2}{n*\theta^2}, \quad when\ n\ is\ large. \tag{2}$$

Here $n$ is the number of neighbors and $\theta = \frac{1}{k}$. However, the number of neighbors is difficult to estimate. One solution to estimate $n$ is based on the node density of network.

The optimal transmission probability will allow a maximum discovery rate. To assess the performance of the node discovery algorithm, we calculate the probability that a node $i$ discovers a node $j$ in a given time slot $t$ which is $P_s$ [12]. To achieve the successful node discovery, the node should score a successful transmission in both slots as discussed below.

In the first sub-slot:

• The node $i$ is transmitting in the direction of node $j$ with probability $\frac{\theta}{2*\pi} * P_t$; And

• The node $j$ is listening in the direction of node $i$ with probability $\frac{\theta}{2*\pi} * P_t$; And

• No other node is transmitting in the direction of $j$. This would take place if other neighbor node(s) transmit in another direction with probability $(1 - \frac{\theta}{2*\pi}) * P_t$ or the neighbor listens with probability $\frac{\theta}{2*\pi} * P_t$.

In the second sub-slot:

• The node $i$ is listening in the same direction with probability 1; And

• The node $j$ is transmitting an acknowledgment in the same direction with probability 1.

The probability of a node $i$ successfully discovering a node $j$ in a time slot $t$ is given by,

$$P_s = 2 * \theta^2 * P_t * (1 - P_t) \times [P_t * (1 - \theta^2) + (1 - P_t) * (1 - P_t * \theta^2)]^{n-1}. \quad (3)$$

We first plot $P_s$ for a beam-width of $\frac{\pi}{8}$ and increasing number of neighbors. The results in Figure 4 clearly indicate that choosing $P_t = \frac{1}{2}$ achieves the maximum neighbor discovery probability. The results for different numbers of sectors also gave similar results (not shown here). Thus, we consider it reasonable to fix the transmission probability to $\frac{1}{2}$ in all our simulations.

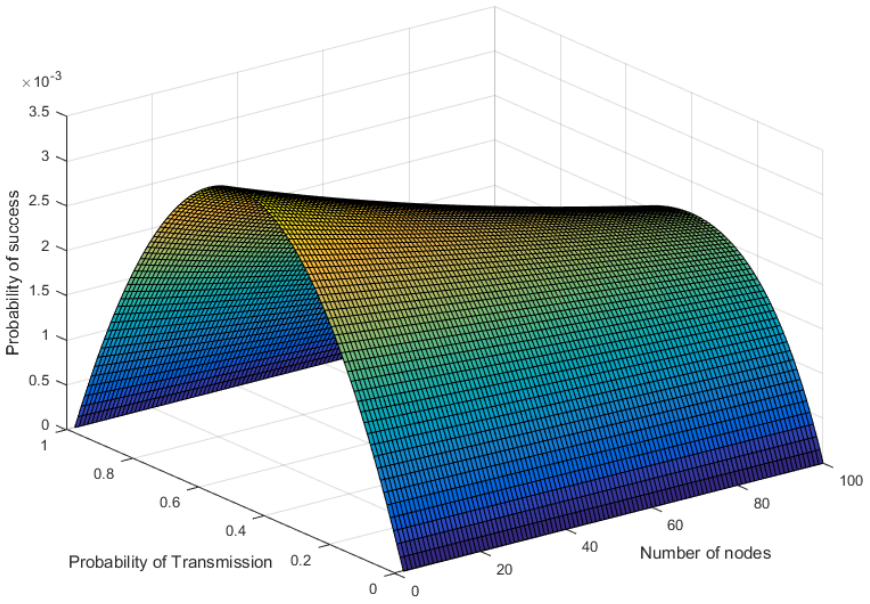


Figure 4. Probability of success for directional antennas with 8 beams.

#### 3.1.3.2. Size of $CW$

To accelerate the neighbor discovery process and avoid nodes colliding again during the collision resolution phase, we maximize the probability that exactly one node selects a given slot to retransmit. We denote the probability that exactly $i$ nodes choose the same slot for re-transmission as $P_s(i)$. Considering that the number of nodes $i$ is a binomial distribution with parameters $(n, \frac{1}{CW})$, we find:

$$P_s(i) = \binom{n}{i} \left(\frac{1}{CW}\right)^i \left(1 - \frac{1}{CW}\right)^{n-i}. \quad (4)$$

We calculate the probability that only one node chooses a given slot to retransmit as :

$$P_s(1) = \left(\frac{n}{CW}\right) \left(1 - \frac{1}{CW}\right)^{n-1}. \quad (5)$$

Differentiating $P_s(1)$ with respect to $n$, we get,

$$\frac{\partial P_s(1)}{\partial CW} = \frac{n(n-CW)(1-\frac{1}{CW})^n}{CW(CW-1)^2}. \quad (6)$$

To find the optimal $CW$ that maximizes $P_s(1)$, let

$$\frac{\partial P_s(i)}{\partial x} = 0. \quad (7)$$

The optimal $CW$ is obtained as:

$$CW = n. \quad (8)$$

## 3.2. Learning Automaton Based Directional Neighbor Discovery Scheme

### 3.2.1. System Model

We consider a network of $N$ randomly deployed nodes. The first step for each node is to detect its one-hop neighbors. The nodes also need to regularly update their neighbor table either through polling mechanism [21] (for mobile networks) or neighbor re-discovery [26] (for static networks). Since the neighbor rediscovery is expensive due to frequent topology changes in mobile networks, the polling mechanism can be used to track the node mobility which would make it easy to compute in which sector the node would move, and update the neighbor table accordingly. On the other hand, since the topology does not change much in static networks, the neighbor rediscovery can be carried out at a very low cost. Each node is equipped with a steerable directional antenna that enables the node to transmit or listen in one of the $K$ non-overlapping sectors. All nodes in the network are clock synchronized. The proposed algorithm operates in an ALOHA-like structure [12, 13]. In each time slot, a node decides to transmit or receive with equal probability. To perform two-way handshaking, each time slot is logically divided into two sub-slots. If a node receives a pilot tone (i.e, sender's ID) in the first sub-slot, it transmits its ID in the same direction (i.e, on the same beam) in the second sub-slot. Similarly, a node in transmitting mode in the first sub-slot

switches to receive mode in the second sub-slot. If two or more nodes transmit towards the same beam of a receiving node, the packets at the receiver are considered lost due to collision.

### 3.2.2. Neighbor Discovery as a Finite State-Action Learning Automaton

A learning environment is referred to as non-stationary if the reward distribution is time dependent. In other words, the objective of an agent in a non-stationary learning environment is not to converge to a single optimal action, but to adjust its behavior in each time slot. In this scheme, the finite-state learning automaton (FLA) is used to model an agent for neighbor discovery, which operates in a non-stationary environment with unknown dynamics [25]. In each time slot, the automaton chooses an action (from a finite set of actions), and receives a *reinforcement signal* from the environment, describing if the selected action is favorable (a reward) or unfavorable (a penalty). Based on the observation, the FLA updates its action selection probability to optimize its future behavior. Here, the objective of each node is to adjust its behavior in each time slot to maximize the discovery rate without converging to a single sector. As more neighbors are discovered in a sector, the optimal policy (i.e, the sector to be searched for neighbors) changes. As no prior policy coordination is assumed, the environment dynamics (i.e, the transition probability) is considered unknown to the agents.

#### 3.2.2.1. The Learning Node

We consider each node $i$ as an intelligent agent empowered by an independent learning mechanism, without prior information about the other agents. The FLA defines the learning agent as a quadruple $\{A_i, \beta_i, P_i, T\}$ [25].

##### 3.2.2.1.1. The Automaton Action Space, $A_i$

An action $a_k$ corresponds to the node choosing a sector $k$ to operate in a listening or transmit mode. $A_i$ is expressed as:

$$A_i = \{a_1, a_2, \dots, a_K\} \ with \ |A_i| = K,$$

where $K$ is the number of sectors.

##### 3.2.2.1.2. The Action Probability Distribution, $P_i$

We allow each node to select a sector probabilistically in a time slot $t$. In particular, each node keeps a probability distribution over the set of available possible actions in a time slot $t$ given by:

$$P_i(t) = \{p_i^1(t), p_i^2(t), \dots, p_i^K(t)\}$$

where $p_i^k(t)$ corresponds to the probability of node $i$ selecting a sector $k$ in time slot $t$. The action probability distribution must satisfy:

$$\sum_{k=1}^{K} p_i^k(t) = 1, \forall t \ , \forall i.$$

#### 3.2.2.1.3. The Reinforcement Signal (Reward/Penalty)

If a node observes a collision in a sector, it assumes that more undiscovered nodes exist in that sector. The probability of exploring this sector should be higher in the following time slots, i.e., the sector is rewarded. Similarly, a penalty is applied to a sector in which a neighbor has been discovered, so that other sectors could be selected instead of selecting this sector again. Thus, the reinforcement learning signal, received by node $i$ in a time slot $t$ is given by:

$$\beta_i(t) = \begin{cases} 0, if\, a\ collision\ occurs(Reward) \\ 1,\ if\ a\ neighbor\ is\ discovered(Penalty) \\ 1,\ if\ the observation\ slot\ is\ idle(Penalty) \end{cases} \quad (9)$$

#### 3.2.2.1.4. The Updating Operator, T

Let $T$ denote the updating operator, which is used to adjust the probability distribution according to $\beta_i(t)$ received after an action. We have:

$$P_i(t+1) = T(P_i(t), A_i, \beta_i(t)).$$

### 3.2.2.2. The Environment

The environment is modeled as a triple of parameters $\{\alpha, \beta, c\}$. Here, the action $\alpha_i(t)$ performed by a node $i$ in a time slot is the input to the environment. The output $\beta_i(t)$ allows the node to assess the action performed, as illustrated in Fig. 5. A key factor describing the environment is the penalty and the reward probability for an action $\alpha_i(t)$. We define the penalty probability vector as [25]:

$$c_i(t) = \{c_i^1(t), c_i^2(t), \dots, c_i^K(t)\}, \quad (10)$$

with $c_i(t) = Pr[\beta_i(t) = 1 | \alpha_i(t) = a_i]$.

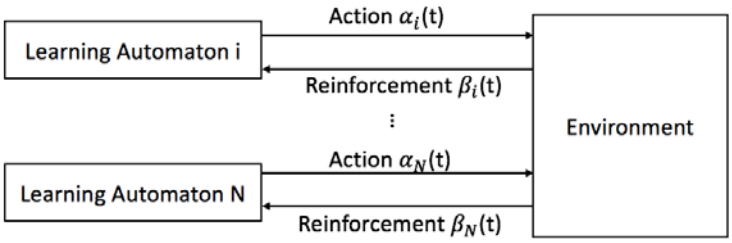


Figure 5. The environment as a set of $N$ independent learning automata.

### 3.2.2.3. The Learning Algorithm

In the existing probabilistic and deterministic neighbor discovery schemes [6, 12], the node operates as a pure chance automaton, i.e., the node does not adjust its behavior based on its current state. By using a learning algorithm, the node can select a most appropriate sector which leads to a successful neighbor discovery with a higher probability. The modification in the node's behavior

is in terms of updating its sector selection probability distribution (from $P_i(t)$ to $P_i(t+1)$) after each time slot, by using the following linear reward and penalty scheme [25]:

- **Initialization:** Each node initializes the action probability vector $P_i(t)$, as $p_i^k(0) = \frac{1}{K}, \forall k, \forall i$. Due to the lack of coordination between the nodes prior to the deployment, the actions are considered equiprobable.
- **Repeat:** At each time instant $t$:
  - *In the first sub-slot:* Select a sector $k \in \{1,2,3,\dots,K\}$ according to the probability vector $P_i(t)$. The transmit or receive operation to perform in that sector is chosen with probability $\frac{1}{2}$.
  - *In the second sub-slot:* The node calculates the reward or penalty, based on the feedback from the environment. Each node independently updates its probability vector by using the linear learning algorithm. For an action $\alpha_i(t)$ that a node $i$ executes in time slot $t$, the probability is updated as follows:

    If $\beta_i(t) = 0 \quad (reward)$:

$$p_i^k(t+1) = \begin{cases} p_i^k(t) + a(1 - p_i^k(t)), if \alpha_i(t) = a_i \\ (1-a)p_i^k(t), if \alpha_i(t) \neq a_i. \end{cases}$$

    If $\beta_i(t) = 1 \quad (penalty)$:

$$p_i^k(t+1) = \begin{cases} (1-b)p_i^k(t), if \alpha_i(t) = a_i \\ \frac{b}{K-1} + (1-b)p_i^k(t), if \alpha_i(t) \neq a_i. \end{cases}$$

    Here $a$ (reward) and $b$ (penalty) are the learning rate, with $a$ and $b \in [0\ 1]$.

## 3.3. Adaptive Directional Neighbor Discovery Scheme

### 3.3.1. System Model and Assumptions

We assume that each node is equipped with a steerable directional antenna with beamwidth $\theta = \frac{2\pi}{K}$, where $0 < \theta < 2\pi$. We denote the neighbors of a node $i$ as $Nei(i)$. Each antenna is able to orient its beam to a predefined direction. To model the directional gain of an antenna, we employ a 2-dimension ideal keyhole antenna model [12]. The beam model takes into consideration only the main lobe, i.e., the direction of maximum power for transmissions or receptions, while the side and back lobes are ignored.

We assume that a pair of nodes can communicate in the directional mode if their antennas are oriented toward each other. In order to formalize this condition along with the presented beam model, we define $\varphi$ and $\varphi'$ as the respective transmit and receive directions of the two nodes. We verify the previous condition through the following formula:

$$\varphi = (\varphi' + \pi) \ mod \ 2\pi$$

We next assume that time is divided into slots and the nodes are perfectly synchronized. The nodes operate in half duplex mode, i.e., a node can either transmit or receive in a given time slot. In transmit mode, a node will send an advertisement message with its ID in a given direction during the first sub-slot. In the second sub-slot, the node will listen for ACK in the same direction. Similarly, a node in listen mode, will dedicate the first sub-slot to receiving potential neighbor discovery messages. If a successful reception occurs and the node is not already discovered, the node will send an ACK message in the same direction in the second sub-slot. If the node is already discovered, the node will remain silent. Moreover, we consider that the nodes possess the following features: Each node is identified by a unique ID; All nodes are perfectly synchronized; The network topology is assumed to be static during the node discovery process.

### 3.3.2. Preliminaries

Various multi-agent reinforcement learning (MARL) schemes have been proposed in literature considering a model-free environment. In this case, the agent ought to learn to behave optimally even when no a priori knowledge about the environment dynamics is available. For the neighbor discovery problem, our goal is to map the directional neighbor discovery problem into a multi-agent learning scenario. To solve this problem, we consider a decentralized implementation of the Q-learning algorithm with an $\varepsilon$-greedy exploration strategy, as it can allow a node to learn an improved policy by repeatedly interacting with the environment with no a priori knowledge of the state transition probabilities. The MARL theory and the Q-learning mechanism are briefly described below.

#### 3.3.2.1. Multi-Agent Reinforcement Learning (MARL)

In distributed decision making, each node is considered as an independent agent with autonomous learning capabilities. Let us assume that the environment is a finite state discrete time non-stationary system. At each time slot $t$, the agent interacts with the system following a predefined sequence:

(1) The agent $i$ observes its state: $s_t^i = s \in S$, where $S$ is the set of $M$ possible states, $S = \{s(1), s(2), \dots, s(M)\}$.

(2) Based on $s_t^i$, the agent $i$ takes an action: $a_t^i = a \in A$, where $A$ is the set of $R$ actions available to the agent: $A = \{a(1), a(2), \dots, a(R)\}$.

(3) A transition occurs as a results of action $a_t^i$. The agent now perceives its state as $s_{t+1}^i$. We denote the transition probability from state $s$ to state $s'$ as $P_{ss'}$.

(4) A reward $r_t^i$ is returned to the agent. The process is repeated infinitely or until an ending condition is met.

The process is depicted in Figure 6.

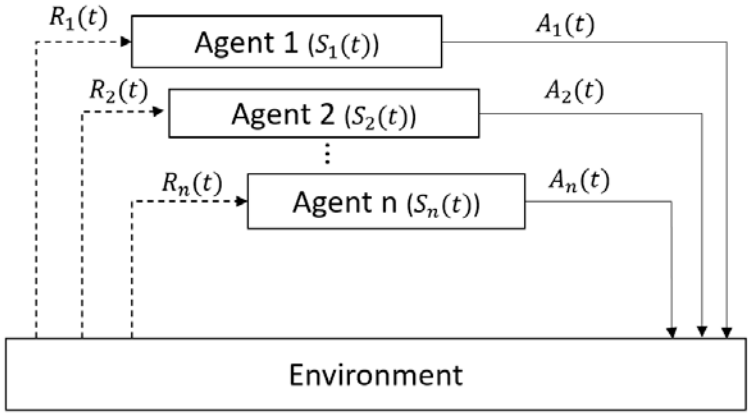


Figure 6. Multi-agent distributed learning scenario.

In MARL scheme, the goal of each agent is to find an optimal policy $\pi^*$ for each state maximizing the reward $r_t^i = r(s_t^i, a_t^i)$ accumulated over an infinite number of steps. We define the value function of a policy as the expected infinite discounted sum of the accumulated reward, when a player starts in state $s_0$ and executes the policy $\pi$.

$$Q(s,a) = \mathbb{E}\{\sum_{t=0}^{\infty} \beta\, r(s_t, \pi(s_t)) | s_0 = s\} \tag{11}$$

where $\beta$ is a discount learning factor $0 \leq \beta \leq 1$. Equation 11 can be rewritten as [27, 28]:

$$Q(s,a) = r(s,a) + \beta \sum_{v \in S} P_{sv}(a) Q(v,b) \tag{12}$$

where, $b \in A$ as $b \notin a$.

Finding an optimal policy is equivalent to minimizing the cost at each time slot:

$$V^*(s) = \min_{a \in A} Q(s,a) \tag{13}$$

By using the Bellman optimality criterion, the optimal policy $\pi^*$satisfies the following equation:

$$Q(s,a) = r(s,a) + \beta \sum_{v \in S} P_{sv}(a) \min_{b \in A} Q^*(s,b). \tag{14}$$

Subsequently, at each time slot a node determines its action as it verifies:

$$Q^*(s,a^*) = \min_{a \in A} Q^*(s,a) \tag{15}$$

As in Equation 14, the Q-value is clearly a function of the excepted reward $r(s,a)$ and the transition probability $P_{ss\prime}$. However, in model free RL problems, the agents don't have *a priori* knowledge of the system dynamics as it generally depends on the others agents' policies. In this case, Q-learning is particularly interesting as it enables a learner to determine an optimal policy by incrementally calculating an estimation of $Q^*(s,a)$, which makes it suitable for our application [27].

#### 3.3.2.2. Q-Learning

Similar to the single agent Q-learning, a multiagent Q-learning scheme operates as follows: each agent maintains an evaluation matrix, denoted by $Q(M, R)$. The Q-learning process estimates the value function in Equation 12 iteratively without any explicit knowledge about the reward $r$ and the transition probability $P_{ss'}$. After execution of each action, each agent independently updates its Q-matrix in a recursive manner [28]:

$$Q_{(s,a)} = \begin{cases} Q_{(s,a)} + \beta \Delta Q_{(s,a)}, if a = a_t \; and \; s = s_t \\ Q_{(s,a)}, otherwise, \end{cases} \tag{16}$$

where

$$\Delta Q(s,a) = \{r_t + \beta \min_{a \in A} Q^{\pi}(v,a)\} - Q(s,a) \tag{17}$$

For the case of non-stationary reward, the Q-learning algorithm needs to keep trying all the action-state combinations as the identity of the optimal action may be different in each time slot [29, 30]. However, the learning algorithm may get stuck in a non-optimal policy unless all the possible actions are tested continuously. To remedy this, an $\varepsilon$-greedy exploration strategy is used, where the agent chooses the greedy action (i.e., the learned action) with probability $\varepsilon$ and a random action with probability $1 - \varepsilon$ [29]:

$$a^* = \begin{cases} argmax(Q(s,a)), & with \; probability \; 1 - \varepsilon \\ I(A), & with probability \; \varepsilon. \end{cases} \tag{18}$$

where I(A) is a characteristic function for the event when action $a$ is chosen. Choosing the numerical value of the exploration rate ($\varepsilon$) should ensure a balance between exploiting the learned policy and exploring new policies to avoid being trapped in a sub-optimal policy. This value is generally chosen empirically.

### 3.3.3. Deterministic Q-Learning based Neighbor Discovery Algorithm

We model the neighbor discovery problem with a directional antenna as an *N-agent model-free learning game*. It is very complex to compute the state transition probability $P_{ss'}$ for the given system model: the reward/penalty received by each agent depends on the strategy of other agents which complicates the estimation of $P_{ss'}$. Further, in a decentralized reinforcement learning scheme, each agent is equipped with an independent Q-learning mechanism which greatly improves the scalability of the proposed scheme compared to the centralized schemes.

As discussed earlier, in the previously proposed schemes, both the beam direction and the operating mode (i.e., trasmit or receive) are determined either randomly or deterministically. In this scheme, we propose a learning formulation based on one dimensional decision making. In the proposed deterministic Q-learning based algorithm, each node executes a scan sequence randomly, and the operating mode is decided by a Q-learning mechanism.

### 3.3.3.1. Deterministic Learning based DND

#### 3.3.3.1.1. Formulation

Each node decides to start a scan sequence in a random direction with probability $\frac{1}{K}$. The operating mode is chosen via the learning algorithm. First, we formulate the Q-learning parameters as follow:

- *State space:* As described earlier, we consider a network consisting of $N$ nodes each equipped with a $K$ sectored antenna. Under a distributed multi-agent framework, each node $i$ focuses on its operating mode and perceives its state at a time slot $t$ as:

$$s_t^i = \{x_i^1(t), x_i^2(t), \ldots . x_i^K(t)\}$$

where $s_t^i$ is a vector of the number of neighbors discovered in sector $k$ after $t$ time slots ($x_i^k(t)$). Thus, considering that the nodes distribution is uniform, it is straightforward that:

$$x_i^k(t) \in \{0, \ldots, \frac{Nei(i)}{K}\} \ \forall k, \forall n.$$

We denote the set of all possible states by $S$. Further, we define a terminating state as a state where a node has discovered all its one-hop neighbors:

$$\mathcal{S}_{term} = s_t^i \in \mathcal{S}: s_t^i = \{\frac{Nei(i)}{K}, \ldots, \frac{Nei(i)}{K}\}\}, \forall i. \quad (19)$$

- *Actions:* After observing its state at each stage, each node needs to independently choose its action for the current time slot. We consider that a node choosing to listen or to receive in a given sector is viewed as an action. The action space of a node $i$ is defined as:

$$A = \{L(listen), T(transmit)\} \quad (20)$$

- *The Reinforcement Signal (Reward/Penalty):* The reward perceived by a node $i$ depends on the observation made in the second sub-slot. The design of the reward function is meant to force the agent to elicit the desired behavior. In the context of neighbor discovery, the agent aims to discover all of its neighbors in a short period. Thus, the agent needs to perform more exploration in the sector where an undiscovered neighbor is expected with a high probability. If a node transmits its ID in the first slot and observes a collision in the second sub-slot, it assumes that more undiscovered nodes exist in that sector. Thus, the Q-value of this sector should be higher in the next time slots. Similarly, when a neighbor is discovered successfully, the reward function should reflect the need to investigate other directions. The reinforcement signal, denoted as $r(s, a)$, received by node $i$ at a given time slot $t$ is given by:

$$r(s,a) = \begin{cases} 1, \ if \ a \ collision \ occurs \\ 0, \ if \ a \ neighbor \ is \ discovered \\ 0, \ if \ the \ observation \ slot \ is \ idle. \end{cases} \quad (21)$$

#### 3.3.3.1.2. Q-Learning Implementation

Having specified the state, action and cost of our formulation, we implement a Q-learning mechanism, where each node executes the following learning steps:

1. Each node initializes its Q-matrix.
2. At each time slot, if a node chooses to act greedily, the selected action is chosen based on the minimum Q-value. If the node chooses to explore, the action is randomly chosen.
3. The node executes the selected action and receives an immediate reward $r(s,a)$.
4. Each node independently updates its Q-table as suggested in Equation 16:

$$Q_{(s,a)} + \beta \Delta Q_{(s,a)} \tag{22}$$

***Algorithm 1: Deterministic Q-Learning based Algorithm***

**Initialization** each $s \in S$, $a \in A$
Initialize the Q-value $Q(s,a) \leftarrow 0$
Evaluate the starting state $s_t^i$ randomly.
**Learning Loop** generate a random number $ran$ between 0 and 1. $ran < \varepsilon$,
Select the start direction randomly,
Select the operating mode minimizing the Q-value.
Execute the selected operating mode.
Calculate the cost based on the reward function in Equation (21).
Observe the next state $s_i^{t+1}$.
Update the Q-matrix entry as in Equation (16).
**End Loop**

Since every node needs to keep looking for new neighbors across all its sectors, the reward is considered as non stationary. In other words, identity of the optimal action (i.e., optimal sector) can be different in each time slot. Thus, the learning procedure needs to find a suitable balance between exploration and exploitation. In this context, exploitation refers to operating in a sector which appears to have the most probable undiscovered neighbors, whereas the exploration refers to choosing any other sector randomly. By introducing an exploration factor $\varepsilon$, a node explores all sectors to ensure that a link is established with all of its neighbors. Numerical values of the exploration factor and learning step are determined via numerical simulation. Details of the algorithm are given in Algorithm 1.

## 3.4. Hardware Demonstration of Directional Neighbor Discovery Scheme

In this section, we describe our hardware demos for directional neighbor discovery (DND) schemes. To better demonstrate the different cases of directional neighbor discovery, we use two directional antennas (instead of one) in each PC. Each antenna is manually rotated to emulate the *steerable* antenna. Manual rotation helps us to better observe the transition process from a 'no-neighbor' case to a 'neighbor detected" case. Our testbed consists of multiple NI USRP 2943R boards for DND test, and we have written the C++ socket programs to achieve multi-point RF communications.

In our DND testbed, each node is equipped with two directional antennas. These two beams are able to send or receive data simultaneously after we program the two Rx/Tx ports of the USRP board. The neighbor discovery in our demos involves node authentication process and multimedia communications.

In the first demo we have implemented the basic directional neighbor discovery via the Rx port programming in the USRP boards. In Demo 2, we show how each node can discover multiple neighbors simultaneously through it two beams. In the third demo, we further test the link switch operations based on different link conditions, in which our intelligent wireless nodes can measure the quality of wireless signal. When the RF signal quality is poor, the sender and receiver automatically switch to another link for a better transmission quality.

### 3.4.1. Hardware Platform

We have used NI USRP 2943R, with 2x2 MIMO RF transceivers with independently tunable operating frequencies from 50 MHz to 6 GHz, with 40 MHz, 120 MHz, or 160 MHz per channel for real-time communication, as shown in Figure 7. The 2x2 MIMO RF transceivers enable the setup of two RX/TX ports and two RX ports. The RX/TX ports can be manually switched between reception (Rx) and transmission (Tx) mode, while the RX ports can only work in the reception mode. The 2x2 MIMO RF transceivers are associated with four individual ADC and DAC chips respectively.

A FPGA chip is the core of the USRP device. It can be used to process RF signals and manage the functionalities of the USRP board. Through the software tool called Labview, a user can change the structure of signal processing sequences. Hence, the USRP supports a flexible wireless communication system with customized wireless protocols.

The directional antenna (HAO14SDP Hi-Gain Antenna) used in our demo has a beam width of 60 degrees. Its operation frequency is 500MHz - 3GHz, with gain of 14dBi, and the impedance of 50 Ohm.

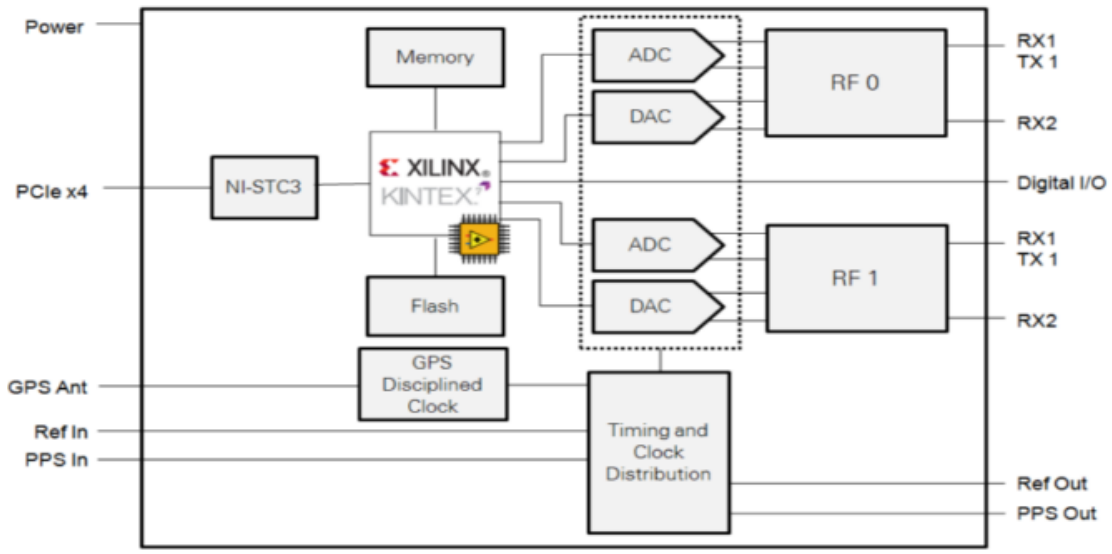


Figure 7. USRP hardware architecture for 2 different RF channels [from NI.com].

### 3.4.2. **Software Platform**

- NI Labview Communication Suite: LabVIEW is used to interact with the FPGA of USRP. The default configuration of NI Labview tool is shown in Figure 8. The Communication Suite of NI Labview can be generally categorized into two components. The first component, shown in the leftmost part of Fig. 8, is the software installed in the host PC. In this component, a user can easily build the wireless communication protocols in the host code and set the communication parameters. In addition, the embedded codes are available in the FPGA chip, which are critical for achieving flexible wireless communication control such as modulation mode change.

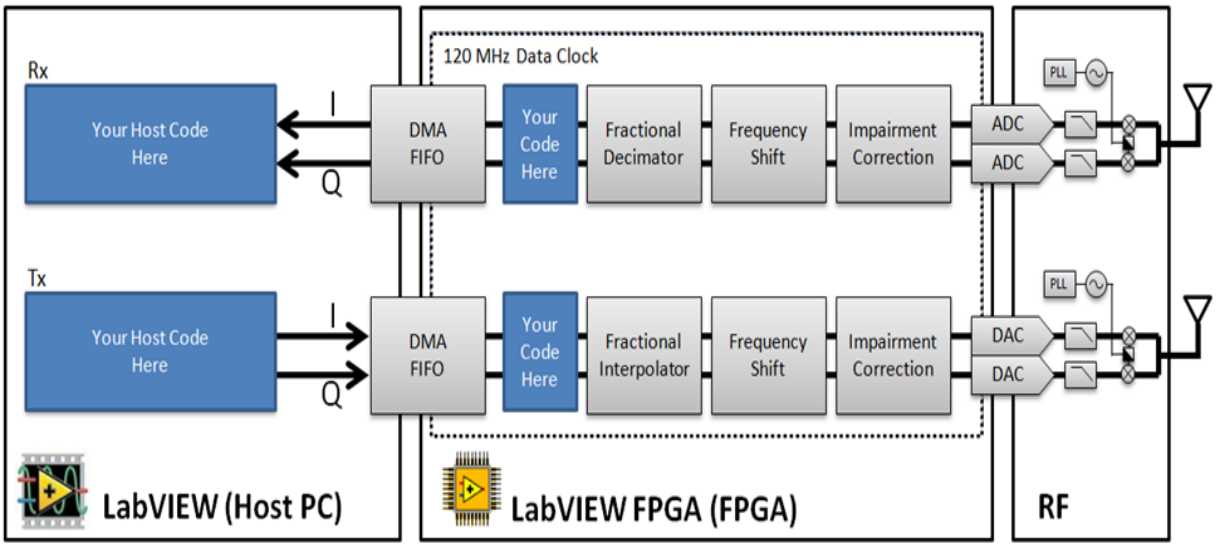


Figure 8. Default FPGA configuration with two RFs [from NI.com]

Users can modify certain part of the codes in the FPGA chip, but it is more complicated than modifying the codes in the host PC. In our demos, since we need to change some functionalities of the USRP board in the lower layer of ISO architecture, we have spent much time building the codes of FPGA in USRP for the implementation of DND scheme.

- C++ Socket Program: We have built a server routine based on C++ sockets to handle the neighbor authentication and channel selection (see Figure 9).

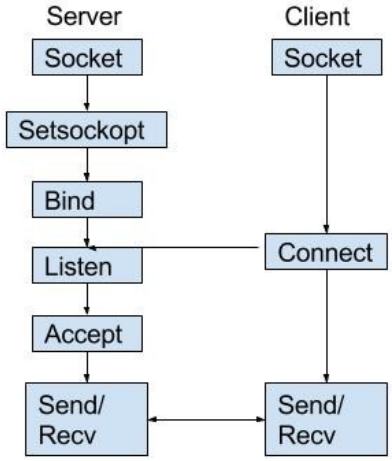


Figure 9. Default FPGA configuration for two RFs [from NI.com].

Socket programming is a way of connecting two nodes on a network for TCP/IP-based communications. One socket (in a node) listens on a particular port with an IP address, while the other socket reaches out to this node to form a TCP connection. In our case, the server forms the listener socket while the client reaches out to the server. Most inter-process communication uses the client/server model.

### 3.4.3. Demo 1: Basic Directional Neighbor Discovery Functions

#### 3.4.3.1. System Model

We build demo 1 by using three USRP-Rio boards (see Figure 10): two as receivers (Rx1 and Rx2), and one as a sender (Tx), at a transmission frequency of 2.2 GHz. Two directional antennas are used in each USRP node. We assume that each node maintains a list of the node IDs of all *valid* neighbors, so that it can decide to accept or decline the communication request from its neighbors based on their 'friendly (valid)' or 'enemy (invalid)' status. In this demo, Rx1 is assumed to be valid whereas Rx2 is invalid. The directional antenna is manually rotated in order to detect RF signals from different directions.

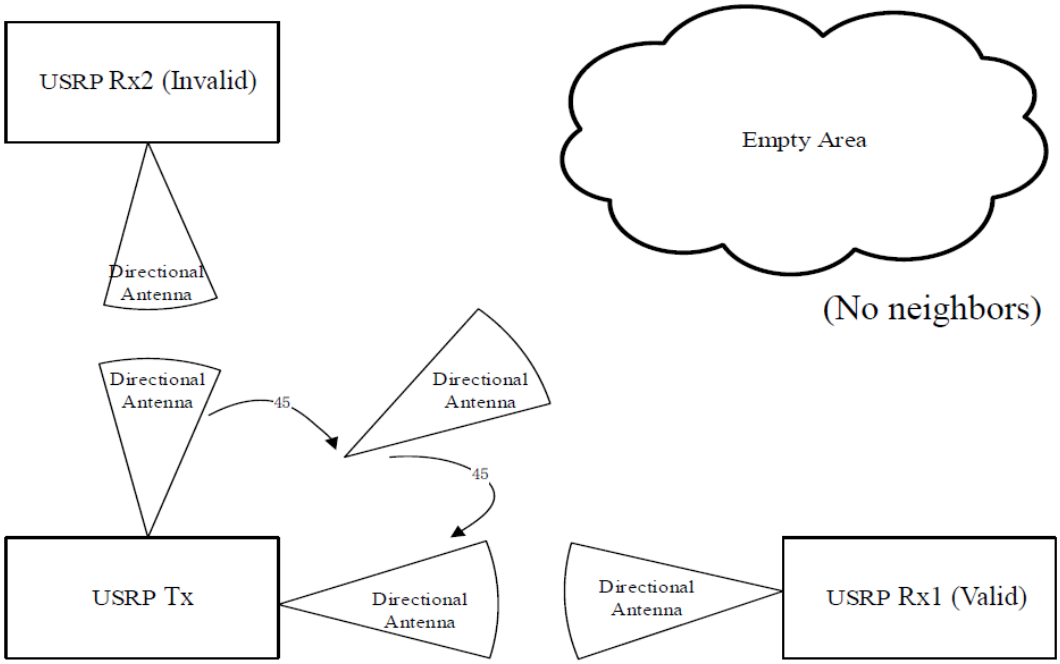


Figure 10. System model of Demo 1.

### 3.4.3.2. Methodology

This demo is used to demonstrate the basic principle of directional neighbor discovery. To mimic the steerable antenna, we manually rotate the direction of a beam. The sender sends a request message through its directional antenna and waits for the response message from its potential neighbors in that particular direction. When no neighbor is detected in that direction, the system displays a message "No neighbor in the area". Otherwise, it will verify the ID of the detected neighbor by checking its valid-IDs list. If the neighbor passes the validation test, the node will add its IP address to the list of valid neighbors, and display "Find a valid neighbor" on the screen. The node uses C++ routine to authenticate the received packet. If the packet comes from an invalid neighbor, the screen displays "Not a valid neighbor".

### 3.4.3.3. Implementation and Test Results

The antenna of Rx1 node faces the front as shown in Figure 11(a) whereas the antenna of Rx2 node is rotated by $90^0$ (facing the left side) in Figure 11(b).

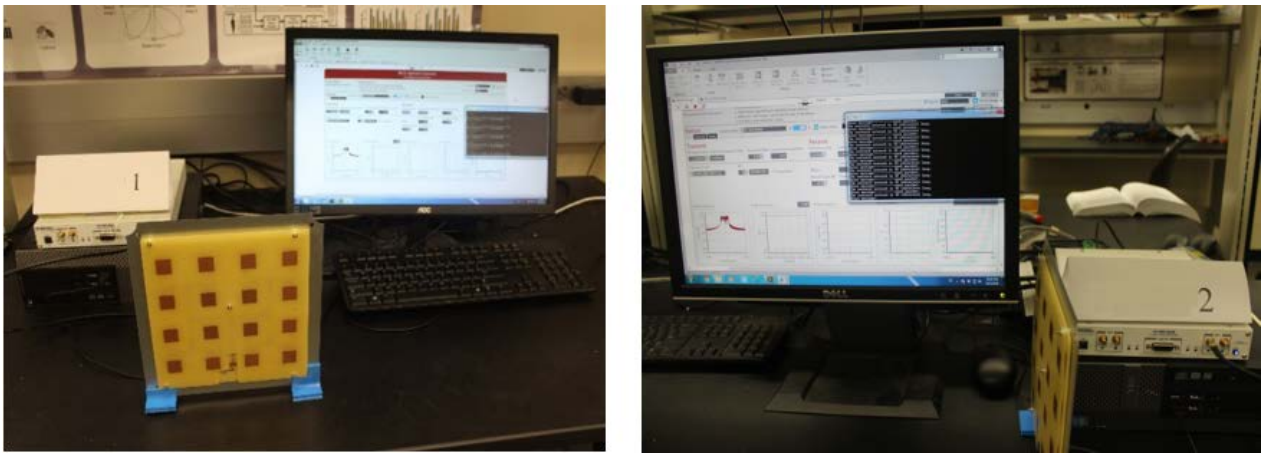


Figure 11. (a) Rx1 setup; (b) Rx2 setup.

Initially the antenna of Tx is set to face Rx2 node and the neighbor discover process is initiated by Tx by sending a “neighbor request” message. The Rx2 node sends a “neighbor response” message back to Tx. However, as mentioned earlier, Rx2 is set as an invalid neighbor of the sender. Therefore, when the sender (Tx) receives the feedback from the receiver (Rx2), it determines that Rx2 is an invalid neighbor, as shown in Figure 12. Thus Tx will not initiate any communication with Rx2.

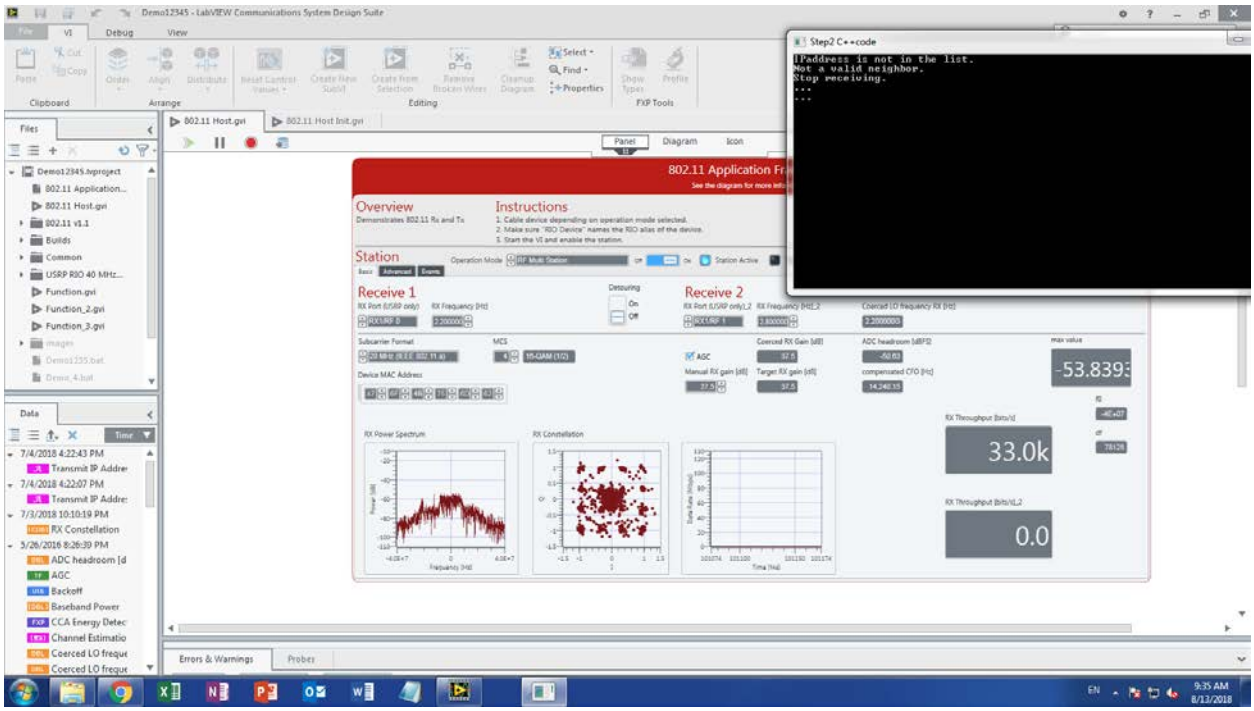


Figure 12. Tx system output while facing Rx2.

In the next stage, the Tx’s antenna is rotated by 45-degree. In this case, its antenna is not oriented toward either of the two receivers, which results in a poor link quality. Since the signal

strength is not strong enough to accomplish the neighbor discovery process, the node (Tx) that is searching for the neighbors cannot find any neighbors. Therefore, we get the message on the screen as “No neighbor in this area”, as shown in Figure 13.

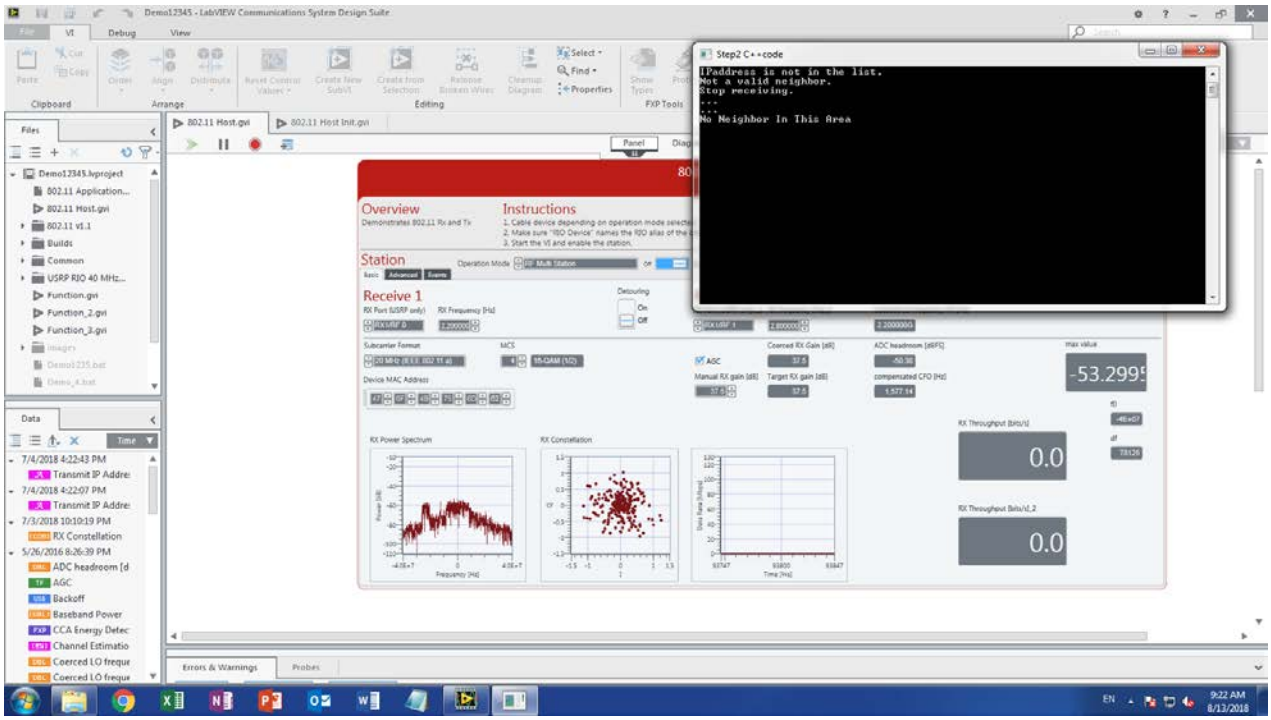


Figure 13. Tx system output when facing an empty area (i.e., without neighbors).

We then rotate the directional antenna of Tx to face Rx1 node, which is assumed to be a valid neighbor. Similar to the previous stages, the Tx node sends the “neighbor request” message on its main beam. Upon receiving this message, the Rx1 node sends back a “neighbor response” message with a valid authentication. In this way, the Tx node determines that the neighbor is a valid node, and adds its IP address to its neighbor list. Now it can communicate with this valid neighbor. Figure 14 shows the result of the successful neighbor discovery process, where the console on the screen outputs the text “Find a valid neighbor”.

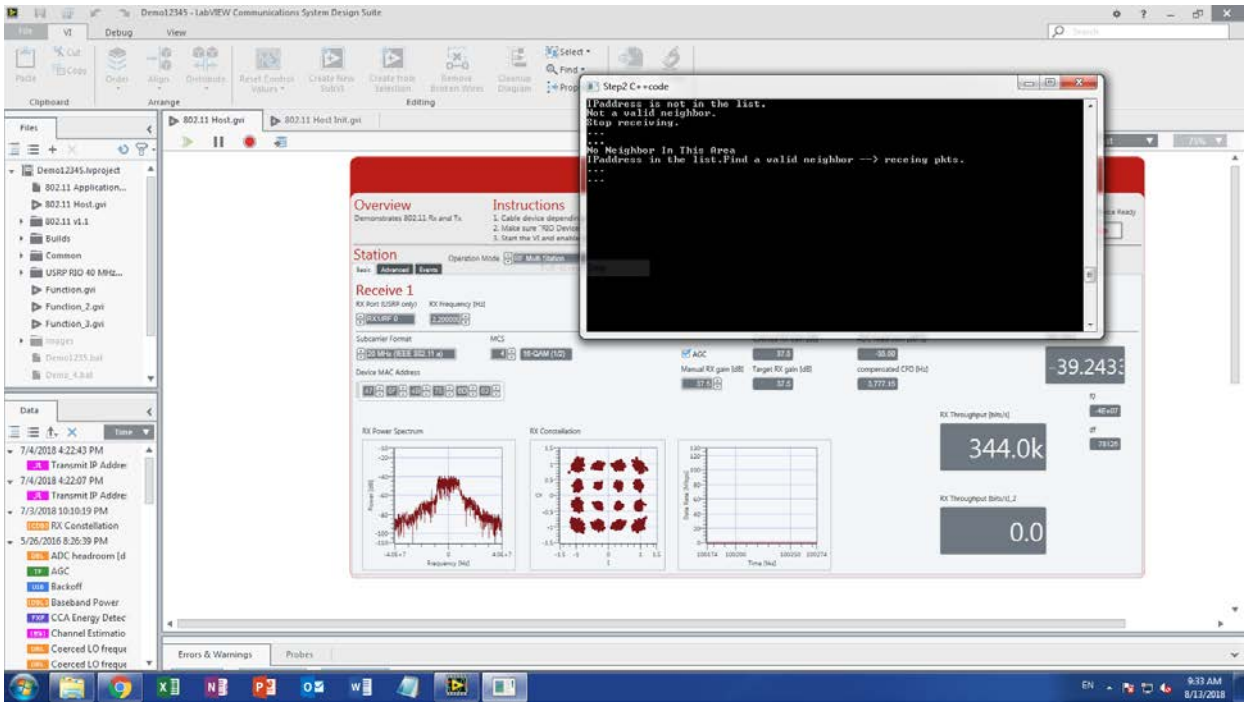


Figure 14. Tx system output when facing Rx1 (valid neighbor).

### 3.4.4. **Demo 2: Multi-Beam Directional Neighbor Discovery**

#### 3.4.4.1. System Model

The USRP-Rio boards are able to transmit/receive data simultaneously through two transceivers (RX1/TX1, and RX2/TX2). Also, the data rate of each transceiver is large enough (>100Mbps) for multimedia transmissions. To explore the advantages of USRPs, we implement the video transmissions in our DND testbed.

As shown in Figure 15, we use three USRPs: one sender (Tx node) and two receivers (Rx1 and Rx2). In this demo, we use two channels with different frequencies, Tx1/Rx1 in Beam 1 at 2.2GHz and Tx2/Rx2 in Beam 2 at 2.8GHz. Two different 720p HD video streams are transmitted by Tx node over these two channels - NCAA Football video stream on channel 1 and National Geographic video stream on channel 2.

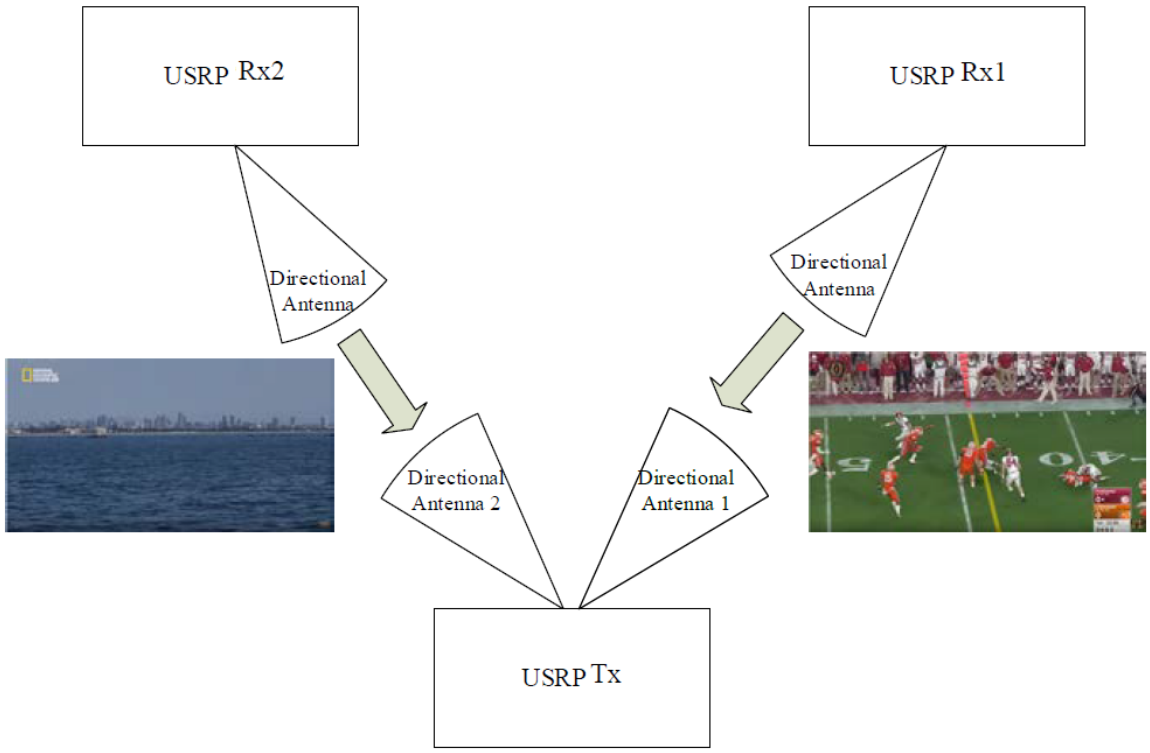


Figure 15. System model for multi-beam neighbor discovery demo.

### 3.4.4.2. Methodology

In this demo, a USRP Tx can discover the neighbors in two different directions. We deploy two USRP nodes (Rx1 and Rx2) in two different directions. Both of them are valid USRP neighbors.

Through careful programming, we make the two antennas work in parallel to discover the neighbors simultaneously. Through multi-beam neighbor discovery process (instead of just relying on one beam), we can significantly shorten the neighbor discovery time of a node.

Note that the USRP board allows the processing of only one Rx data flow and one Tx data flow in its FPGA-supported libraries by default. To enable simultaneous neighbor discovery in both beams, we have implemented another Rx data flow transmission thread in its FPGA board. Since this modification cannot be achieved through a regular programming in the host PC, we had to modify the codes in the FPGA which is very time consuming.

As mentioned before (see Figure 8), the data transmission control of a USRP is programmed through a flow chart management module. A packet is processed by the blocks in the flowchart, where each block carries out a particular data processing function. This procedure and the blocks are defined by the code stored in the FPGA board of USRPs. The USRP manufacturer does not recommend changing this architecture. However, to implement this modification, we obtained a special permission from the manufacture, and realized our design after investing a lot of effort.

### 3.4.4.3. Implementation and Test Results

In this demo, the sender (Tx) has two antennas (i.e., two beams), one beam facing Rx1 and the other facing Rx2. Each beam goes through the same steps as in Demo 1, and the two beams perform those operations simultaneously. After the two neighbors (Rx1 and Rx2) are detected via these two beams, the sender (Tx) begins to transmit multimedia data (video stream) to these two neighbors at the same time.

Figure 16 shows the two video streams that we transmitted through two different beams of Tx node. Due to the high bandwidth supported by USRP boards, both videos are high-resolution streams and can be played on the receivers (Rx1 and Rx2) in real-time. In Figure 16, one can also observe the status of the system through the two consoles at the bottom of the figure.

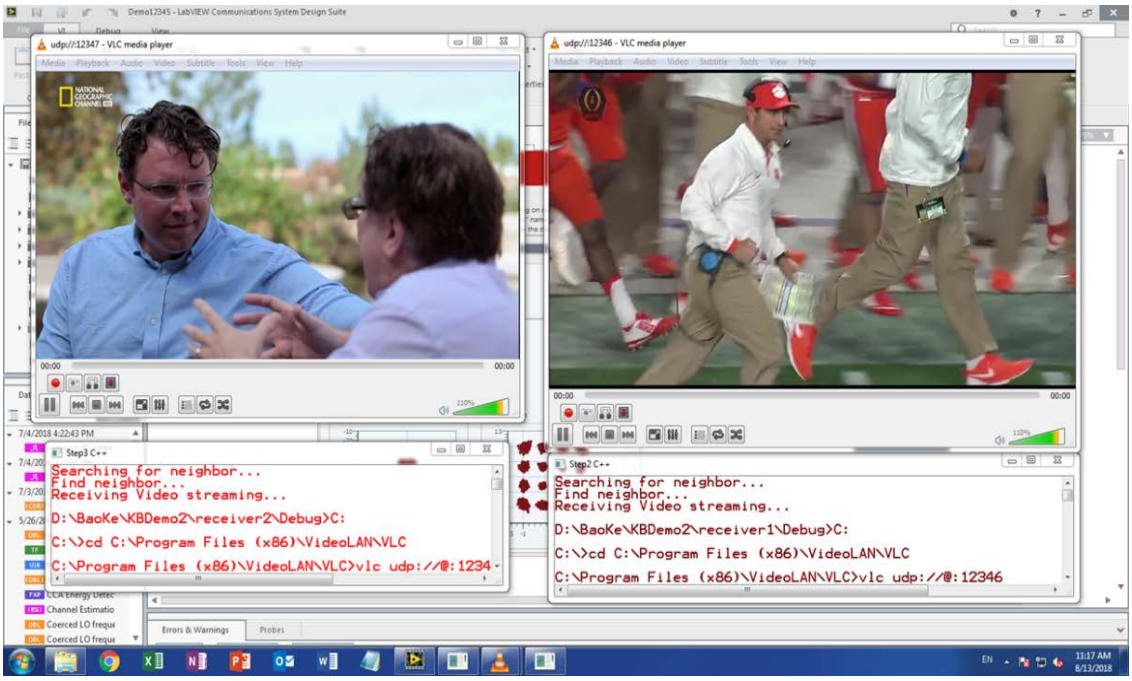


Figure 16. System state when two neighbors are discovered (Left side is the video stream sent to Rx2, Right side is for Rx1).

## 3.4.5. Demo 3: Multi-Beam Link Switching

### 3.4.5.1. System Model

In this demo, we implement the link quality detection function in the DND testbed. The link quality detection can further improve performance of wireless networks by preventing the routing of packets through the neighbors that have poor link quality (measured by the poor received signal strength). We use a two-antenna system in this testbed, which can establish two links with respective neighbors, one link per antenna. In practical networks, the link quality can be actively sensed based on the received signal strength to establish the best end-to-end route among the neighboring nodes. Therefore, our tasks here include basic neighbor discovery and link quality comparisons among neighbors.

The setup of this demo is similar to Demo#2 shown in Figure 15, the major difference being that the video stream is transmitted only on one of the two links, based on the link quality and priority. Here Link 1 (in Antenna 1's orientation) that streams the football video has a higher priority than Link 2.

### 3.4.5.2. Methodology

We demonstrate that the system automatically switches the link based on their quality and priority. Here we use the state variable $S_1 \in (0,1)$ to denote the link signal quality (1: good; 0: poor) and the state variable $S_2 \in (0,1)$ to indicate whether the current link is being used (0: idle; 1: occupied). If Link 1 is in good condition ($S_1 = 1$), $S_2$ is set to 1 to inform that Tx is streaming the video to Rx 1 node on it. Otherwise, the system automatically switches to the other available link (i.e., Link 2 between Tx and Rx2, if it is in good condition, until the condition of Link 1 becomes good again.

### 3.4.5.3. Implementation and Test Results

Initially, both links are in good condition (their signal-to-noise ratio (SNR) is -38dBm, and $S_1 = 1$). Since Rx1 has a higher priority, Tx sends the football video to Rx1 as shown in Figure 17.

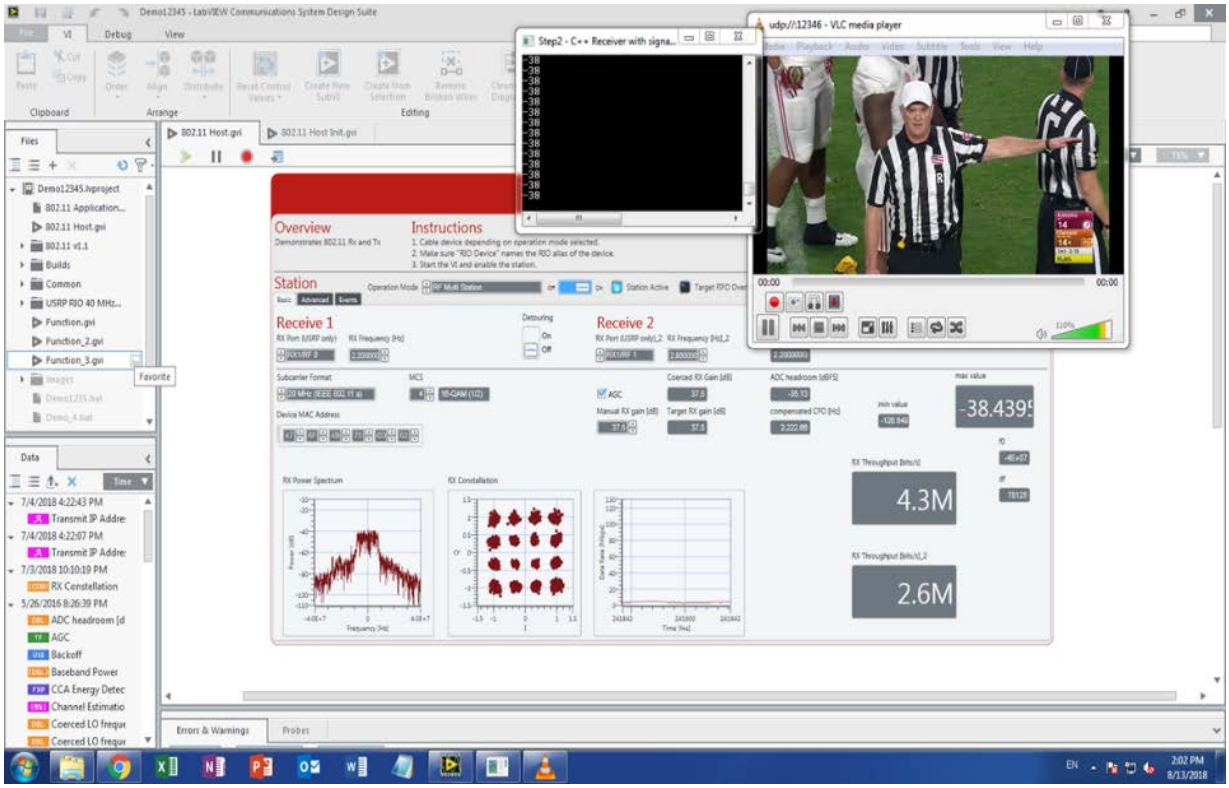


Figure 17. $S_1 = 1$ *and* $S_2 = 1$, Tx sends the football video to Rx1.

Then we block Link 1 by placing an object in its direction, which degrades its signal quality

to poor (its SNR is -51dBm, $S_1 = 0$). As a result, Tx automatically switches to the video stream to Rx2 on Link 2 ($S_2 = 1$), as shown in Figure 18.

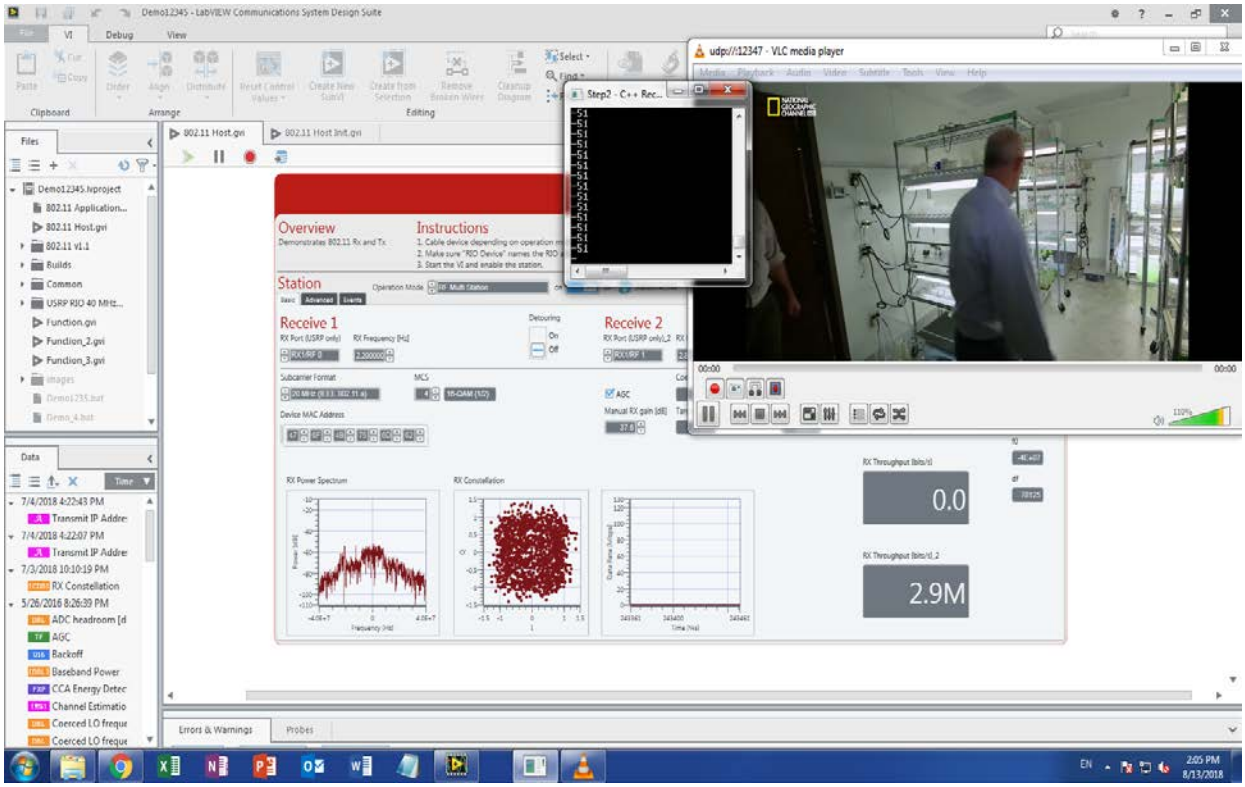


Figure 18. $S_1 = 0$ *and* $S_2 = 1$, Tx sends the video to Rx2.

When we remove the obstacle from Link 1's direction to make its quality "good" again (the signal to noise (SNR) is -38dBm, $S_1 = 1$), the system switches back to stream Rx1 video, as shown in Figure 19. These results demonstrate that Tx can automatically switch between two links based on the traffic priority and channel conditions.

Major advantage of this Demo 3 is that it demonstrates the automatic update of the established routes when more than one neighbor is present which make available multiple links.

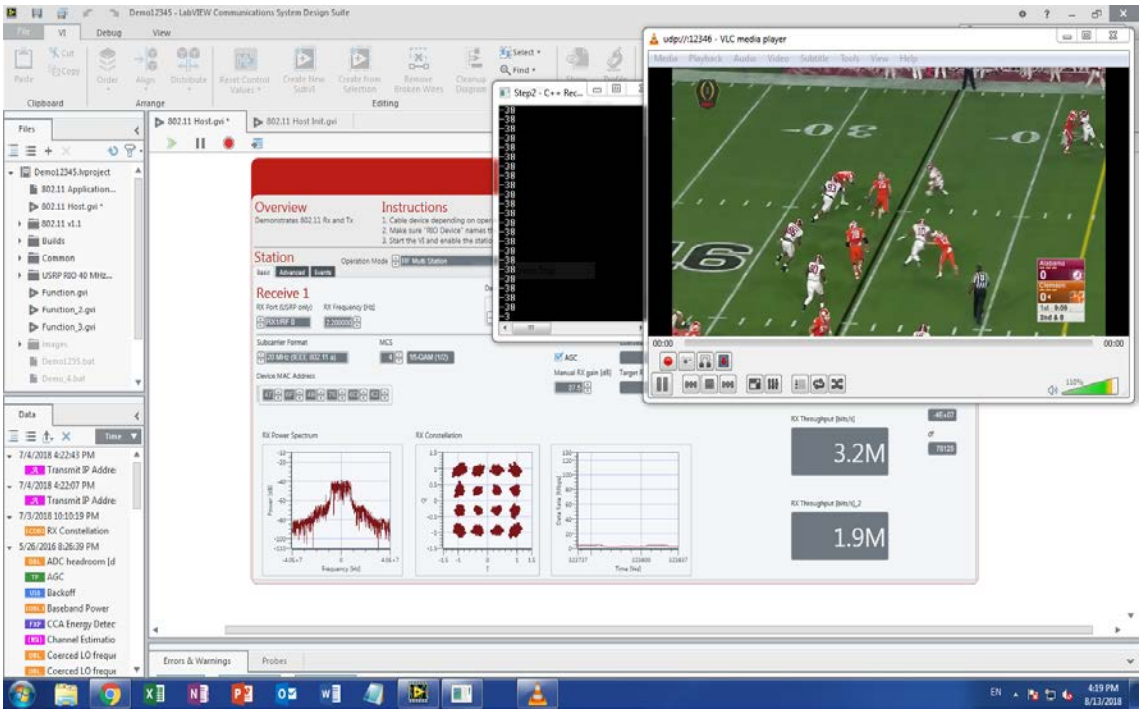


Figure 19. $S_1 = 1$ *and* $S_2 = 0$, Tx sends the Football video to Rx1.

# 4. RESULTS AND DISCUSSIONS

In this chapter, we discuss the performance of our three directional neighbor discovery schemes that we described in Chapter 3.

## 4.1. Collision-Aware Directional Neighbor Discovery Scheme: Performance Analysis

To assess the performance of the proposed scheme, we split the evaluation into two major sections. In the first section, we analyze the improvement of the new algorithm compared to the random and scan schemes proposed in [12] and [31]. We then enhance the achieved results by enabling cooperation between nodes. The peer assisted version of the proposed algorithm operates identically as described in Section 3, the only difference being that a node also includes its neighbor table in the exchanged discovery packets. This will allow a node to discover multiple neighbors indirectly.

The evaluation of the proposed algorithm is performed using extensive simulations using MATLAB. To quantify the improvement achieved by the proposed scheme, we use the following metrics which have been widely used in the literature:

- **Convergence time:** we define the convergence time of the neighbor discovery process as the number of slots required for all the nodes to discover 90% of their neighbors.
- **Discovery ratio over time:** This metric represents the ratio of the discovered neighbors to the total number of neighbors over time.

We also study the performance of the directional neighbor discovery scheme when the network density and the antenna beamwidth vary. The simulation experiments in different scenarios provide a realistic estimate of the time duration required to achieve the network wide neighbor discovery.

### 4.1.1. Unassisted Collision-Aware Neighbor Discovery

First, we compare the performance of the proposed algorithm with random and scan based algorithms in the same simulation environment. The simulation scenario consists of 100 nodes randomly distributed in a square area $5 \times 10^9 \mathrm{m}^2$. Each node has a directional transmission range of 500m and 8 sectors. The results are averaged over 30 runs corresponding to different nodes placements.

Figure 20 shows the ratio of neighbors discovered as a function of time for the unassisted collision-aware algorithm and the random and scan based neighbor discovery schemes without any collision management. The results clearly indicate that introduction of the collision management mechanism allows nodes to discover their neighbors at a faster rate. Quantitatively, the proposed algorithm requires 57% fewer slots to complete the network-wide neighbor discovery compared to the scan-based algorithm.

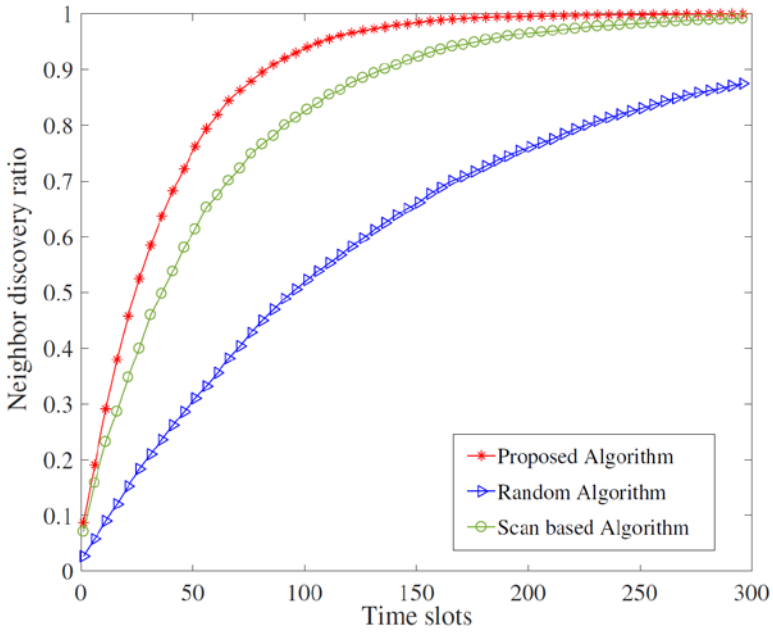


Figure 20. Discovery ratio for the unassisted and assisted neighbor discovery schemes considered scenario($N = 100, k = 8$).

It is also of great interest to evaluate the performance of the proposed scheme for the sparse and densely populated networks. Figure 21 plots the convergence time for the three algorithms with different values of node density. As expected, the number of slots required to achieve the convergence of the neighbor discovery process is proportional to the node density. Moreover, the improvement of the proposed algorithm is also strongly correlated to the node density. For higher number of neighbors, the improvement achieved by our proposed scheme is more prominent. This may easily be substantiated as the number of packet collisions also increases for higher densities.

Next, we investigate the performance of the proposed algorithm for nodes with different number of beams. We fix the number of neighbors to 12. Figure 22 shows the average convergence time when the beamwidth varies from $90^0$ to $25^0$. First, we observe that the random algorithm needs a lot more time to complete the neighbor discovery process for narrow beams. Second, the proposed algorithm outperforms scan based for both the wide and narrow beams.

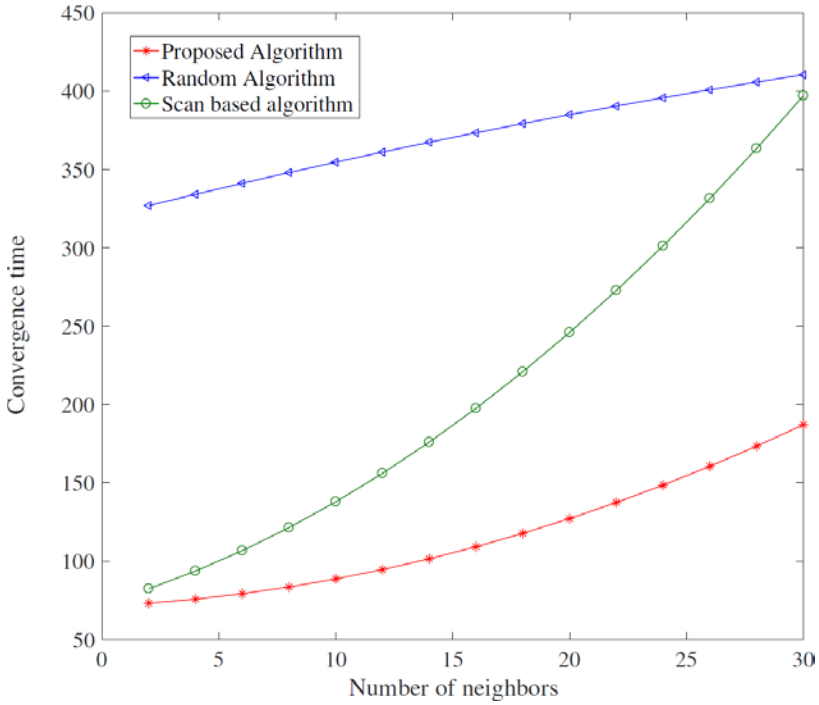


Figure 21. Convergence time for ( K=8) and increasing number of neighbors.

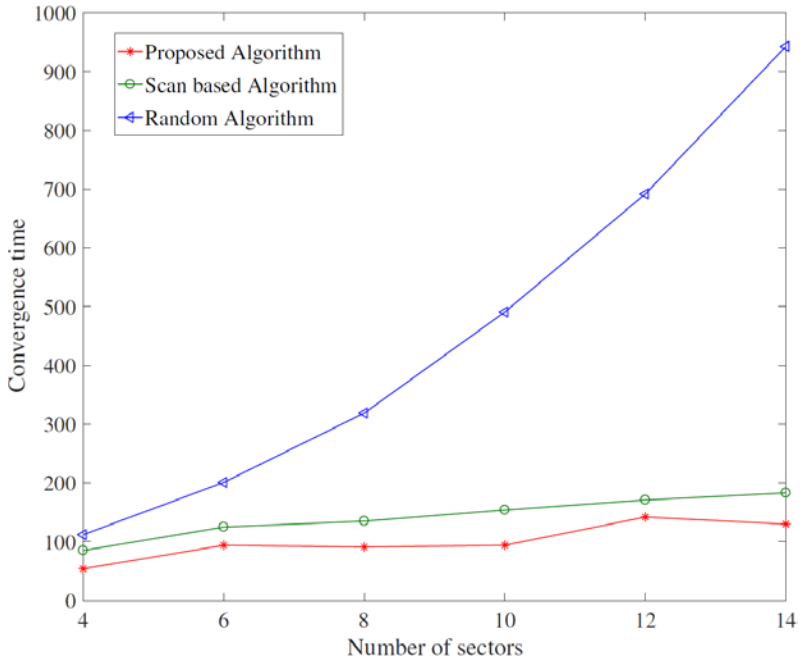


Figure 22. Convergence time for different number of beams and a fixed number of neighbors.

### 4.1.2. **Peer-Assisted Collision-Aware Neighbor Discovery**

We implement an enhanced version of the collision-aware neighbor discovery scheme by enabling the cooperation among nodes. In this version of the algorithm, we consider that all nodes have knowledge of their locations via GPS. Each node includes the *(ID)* and the location of its already discovered neighbors in its discovery packet. This will allow nodes to discover potential neighbors indirectly minimizing the number of slots required to complete the network-wide neighbor discovery. Figure 23 shows the neighbor discovery ratio achieved by the proposed and the scan based algorithms, along with their peer assisted versions. We observe that peer-assisted algorithms outperform the unassisted algorithms in terms of the discovery ratio over time. This is not surprising since, for the peer-assisted algorithms, one successful transmission can allow a node to discover multiple neighbors indirectly.

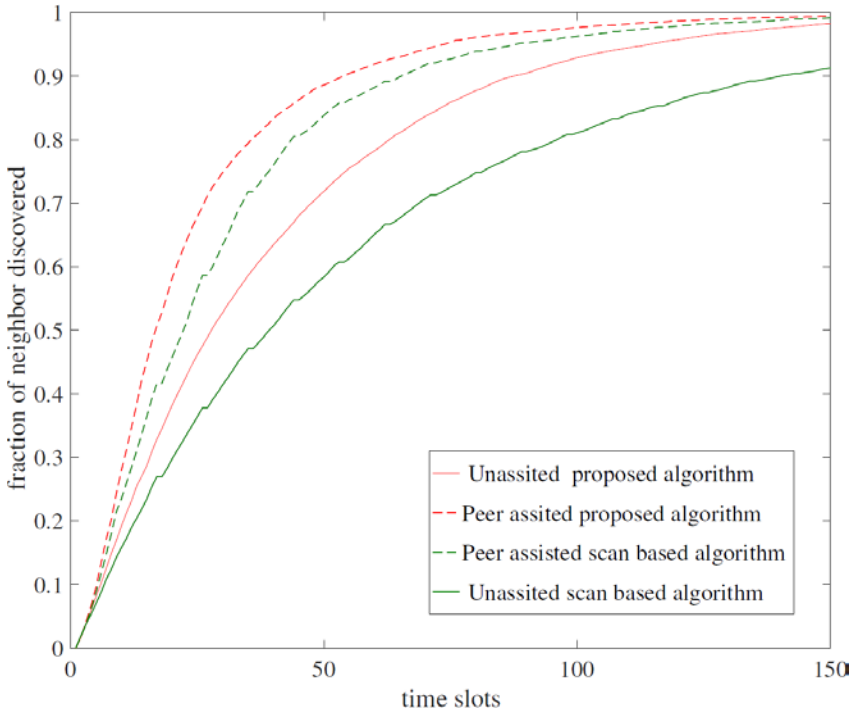


Figure 23. Discovery ratio for the the unassisted and assisted neighbor discovery schemes scenario($N = 100, k = 8$).

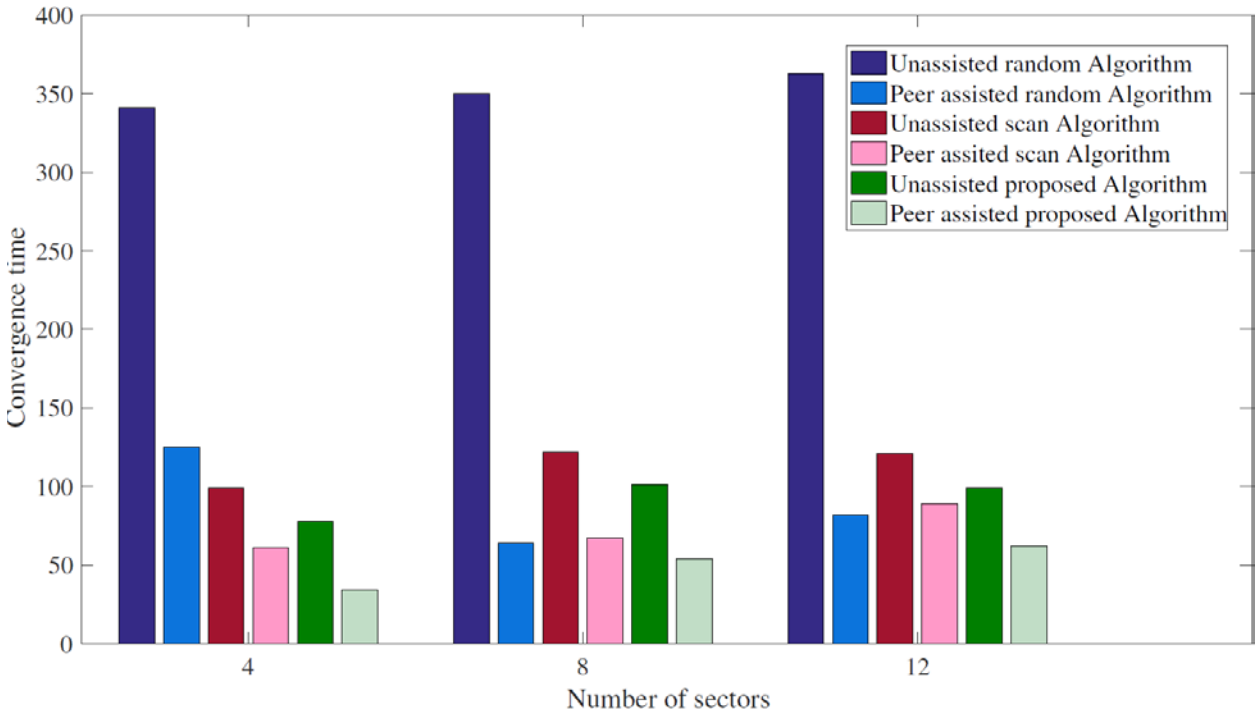


Figure 24. Convergence time for different algorithms with varying number of beams

Next, we study the performance of these schemes for different numbers of antenna sectors. As expected, the peer-assisted collision-aware algorithm significantly outperforms the other schemes as shown in Figure 24. Similar to our previous analysis, we compute the convergence time for networks with increasing density. The results show that the convergence time follows the same trend as the previous results. In summary, the results suggest that combining the proposed collision resolution management and the cooperation between nodes (peer-assisted) achieved significant improvement in terms of reduced neighbor discovery time.

## 4.2. Learning Automaton Based Directional Neighbor Discovery Scheme: Performance Analysis

### 4.2.1. Convergence Behavior

Generally, the aim of a learning scheme is to converge to an optimal action(s) based on past observations. However, the node must keep looking for new neighbors across all sectors during neighbor discovery. In other words, the optimal action (i.e., optimal sector selection) can be different in each time slot. In fact, the variation pattern of the expected reward would give a precise idea about the evolution of the optimal policy. In Figure 25, we plot the probability of observing a reward (represented as the collision probability) as a function of time (see Equation 9). The probability of collecting a reward is initially high, but it decreases with time. This property sustains the claim of the non-stationarity of learning environment as the penalty probability is time dependent, and the agent needs to adapt to the changes without converging to a given action. A

node responds to a neighbor discovery advertisement, only when no previous link has been established with the sender node [13]. Thus, as the node discovers most of its neighbors, the probability of collisions decreases, because the previously discovered nodes remain silent.

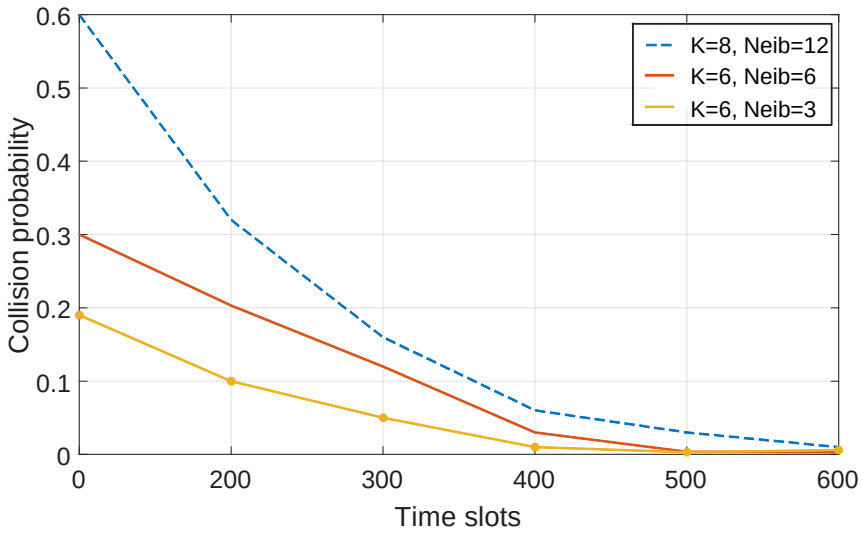


Figure 25. Collision probability for different number of sectors (or beams, $K$) and neighbor density values. Here, Neib is the number of neighbors per beam.

Hence, the asymptotic behavior of LA can be described in terms of the asymptotic behavior of the penalty probability vector (defined in Equation 10) as:

$$\lim_{t\to\infty} c_i^k(t) = 1, \ \forall i, \forall k.$$

Next, we are interested in the final distribution of sector selection probability, $P_i$. To calculate the limiting distribution of $P_i$, we use the expected probability distribution as $E[P_i(t+1)|P_i(t)]$. Using the distance diminishing factor, it can be shown that the limiting distribution of $P_i$ is normal with a mean [25]:

$$E[P_i^k(\infty)] = \frac{c_i^k(t\to\infty)}{\sum_{j=1}^{k} c_j^k(t\to\infty)}, \ \forall i, \forall k.$$

As discussed earlier, the probability of a node receiving a penalty will go to 1 as the neighbor discovery process converges. Thus, the limiting distribution will converge to an equilibrium point as:

$$E[P_i^k(t\to\infty)] \approx \frac{1}{K}, \forall k, \forall i.$$

The interpretation of this result perfectly matches with our specific protocol design requirements. During execution of the proposed strategy, the node redraws its sector selection probability to accomplish a fast neighbor discovery. However, by the end of the algorithm, all the sectors will be equally penalized as no reward is expected. This will lead to an equilibrium point approximately equal to the initial distribution.

### 4.2.2. **Impact of Design Parameters**

Value of the learning parameters $a$ (reward) and $b$ (penalty) dictates the amount of adjustments applied to the action probability vectors $P_i(t)$. Intuitively, the value of parameters depends on the number of sectors and node density. For example, in a high density network, more exploration is needed in a given sector even after a successful neighbor discovery, in order to reach its other neighbors. A small value of the penalty parameter ($b$) is more appropriate in this case, as the variation of the probability vector should be small to allow the exploration of that sector in later slots. The same logic can be used for the reward parameter ($a$). As dense networks are more prone to collisions, a high value of '$a$' will allow the colliding nodes to discover each other in a relatively short period of time by assigning higher probabilities to the sector(s) experiencing collisions. In contrast, sparse networks need less exploration as the number of neighbors per beam is small. A more significant adjustment of $P_i$ (i.e., a high value of $b$ ) allows exploitation of the information collected by investigating other sectors. A good estimate for the update of $a$ and $b$ can be defined through numerical simulations as suggested in [25].

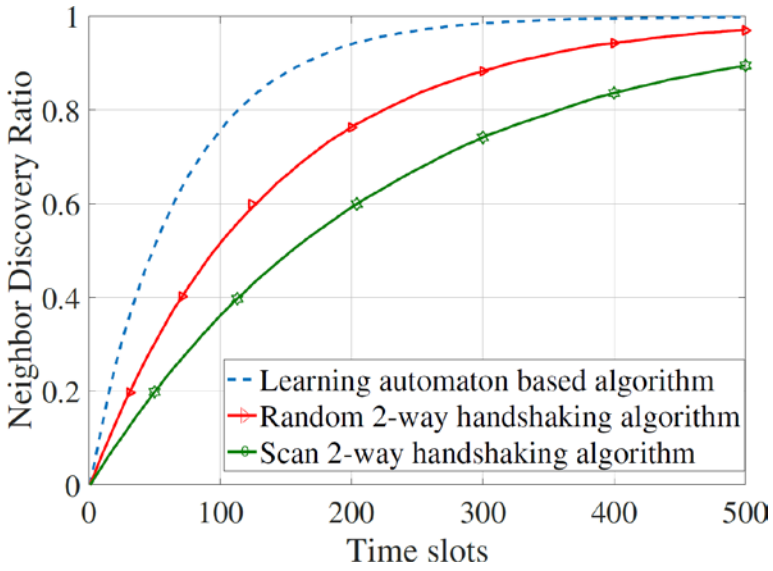


Figure 26. Discovery ratio over time for $K = 8$ and $N = 150$, with learning parameters ($a = 0.1, b = 0.05$).

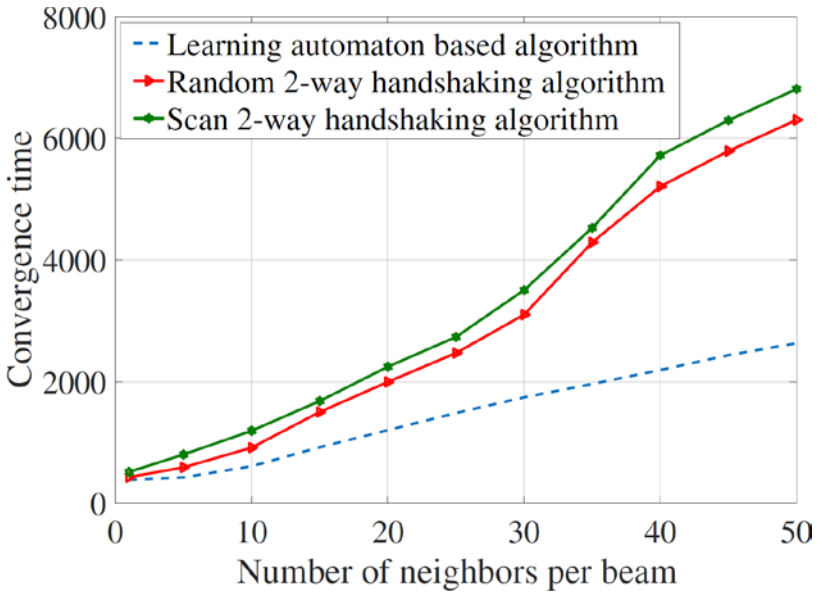


Figure 27. Convergence time for $K = 8$ and increasing neighbor density.

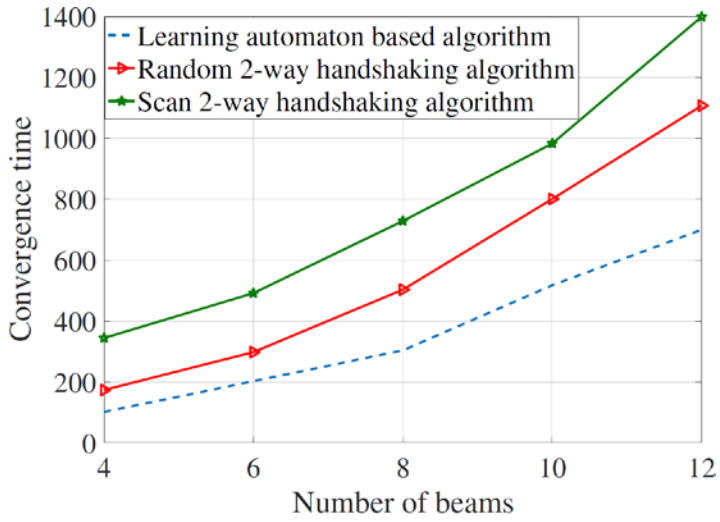


Figure 28. Convergence time for different number of beams with 4 neighbors per beam.

### 4.2.3. **Simulation Results**

In this section, we evaluate performance of the proposed scheme, compared to the two-way random handshaking algorithm [12] and the scan based algorithm [6]. Latency of the neighbor discovery scheme is quantified by two widely used performance metrics: the neighbor discovery ratio over time and convergence time, defined in Section 4.1. Minimizing these metrics also minimizes the energy consumption during the neighbor discovery process.

The simulation scenario consists of 150 nodes randomly distributed in a 1000m by 1000m area. Each node uses 8 antenna beams and has a directional transmission range of 200m. The results are averaged over 30 runs corresponding to different node placements. The neighbor

discovery ratio as a function of time in Figure 26 clearly indicates that the proposed LA based scheme achieves a faster neighbor discovery rate. Quantitatively, the proposed scheme requires 48% fewer time slots to achieve 90% neighbor discovery compared to the random scheme, and 68% fewer slots compared to the scan-based algorithm.

In Figure 27, we compare the convergence time for different node densities per beam. The learning parameters are adjusted to maximize the discovery rate. As expected, all the three schemes need more time to achieve network-wide neighbor discovery for a higher node density. However, the use of a learning based scheme reduces the convergence time considerably. This improvement is more obvious for dense networks because collisions occur more frequently. The learning mechanism can exploit the collision information efficiently, which leads to a faster discovery. On the other hand, the *explore only* strategy of the random and scan-based algorithms results in a significantly increased delay.

The neighbor discovery performance for different antenna beamwidths is evaluated in Figure 28, where the number of neighbor nodes per beam is fixed to 4, and the number of antenna sectors is varied from 4 to 12 (i.e., beamwidth varies from 90˚to 30˚). Although all three schemes need a higher convergence time for narrow beams, the proposed learning-based scheme provides much lower latency.

## 4.3. Adaptive Directional Neighbor Discovery Scheme: Performance Analysis

In this section, performance of the proposed scheme is compared with the probabilistic and deterministic schemes proposed in [12] and [6]. We also consider multiple simulation scenarios to study how the neighbor discovery process is affected by the network characteristics, including the network density and antenna beamwidth. To capture the effectiveness of each protocol quantitatively, we use the average discovery ratio over time and convergence time metrics.

### 4.3.1. Neighbor Discovery Latency

We first evaluate performance of the deterministic Q-learning based scheme compared to the randomized and scan based schemes proposed in [12] and [6]. The simulation scenario consists of 100 nodes randomly deployed in a square area of 1Kmx1Km. Each node is equipped with a steerable antenna with a 60˚ beamwidth and transmission range of $200m$. The average number of neighbors per beam is five. To obtain accurate results, we repeat the simulation experiment 30 times with different node placements, and the results are averaged for each data point. Figure 29 shows the average discovery ratio for the considered scenario. The use of Q-learning scheme significantly reduces the neighbor discovery latency. The deterministic Q-learning based scheme requires 151 and 438 fewer time slots than the random and deterministic 2-way handshaking schemes, respectively, to achieve a 90% network wide neighbor discovery. Further, the Q-learning based scheme significantly reduces the effect of long-tail problem on the discovery latency. As the node develops knowledge about the neighborhood in each sector, it learns to focus its beam in the directions where less neighbors have been discovered instead of searching (randomly or deterministically) in all sectors. Thus, a node significantly reduces its discovery latency by adapting it search probability to focus on the undiscovered neighbors in their respective sectors.

### 4.3.2. Impact of Node Density

Figure 30 shows the average convergence time for different node densities in the network. Average number of nodes per beam is varied from 2 to 10 neighbors for a fixed beamwidth of 60°. For each case, the learning parameter $\beta$ and the exploration parameter $\varepsilon$ are adjusted to maximize the discovery rate. As the node density increases, a node needs more time to establish links with all of its one-hop neighbors due to a higher collision probability and increased deafness. However, the Q-learning algorithm exhibits a significantly lower latency for all the considered scenarios. We also note that the improvement is more prominent when the number of neighbors increases. For the case of a higher density (10 neighbors per beam), the algorithm achieves a network wide neighbor discovery with a reduction of 368 and 581 time slots as compared to the random and deterministic 2-way handshaking schemes, respectively. This can be explained by the fact that the integration of Q-learning mechanism allows a node to adjust its behavior upon a collision or a successful communication. Thus, a balanced exploitation/exploration scheme is particularly helpful in the collision-prone situations, as it can allow a faster collision resolution compared to the continuous exploration used by the random and deterministic schemes.

### 4.3.3. Impact of Antenna Beamwidth

To evaluate the impact of antenna beamwidth on neighbor discovery latency, we performed additional simulations by fixing the number of neighbors per beam to 5 and varying the number of sectors from 4 to 12 (i.e, varying the beamwidth from 90° to 30°). The results are shown in Figure 31. As expected, the use of narrower beams experienced a higher latency. The deterministic Q-learning based algorithm exhibits the shortest convergence time in all the considered scenarios. Moreover, the convergence time for the Q-learning based scheme is significantly lower in the case of narrow beams. Quantitatively, the learning algorithm can achieve up to 28% and 56% faster network wide discovery compared to the random and deterministic schemes, respectively (for $K = 12$).

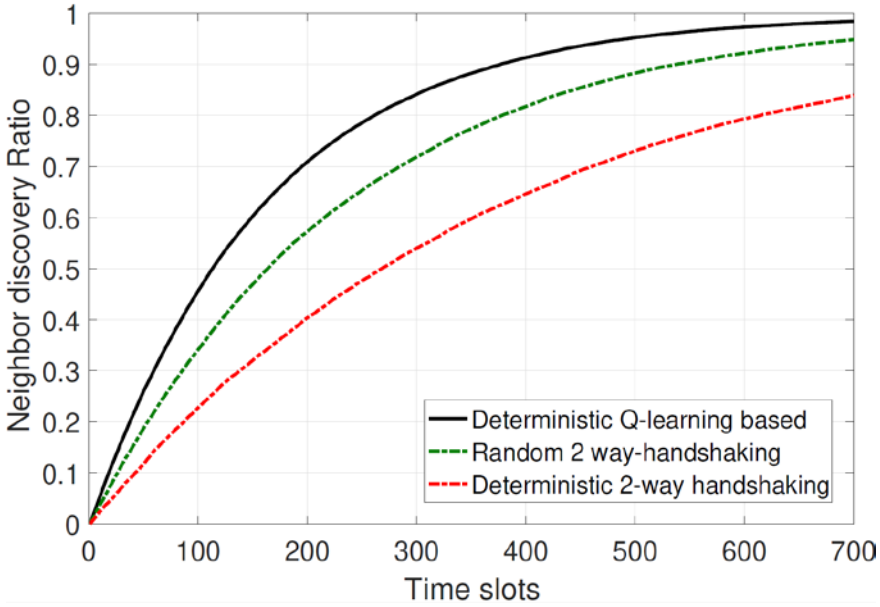


Figure 29. Average discovery ratio for ($Nei = 5$nodes/beam and $k = 6$).

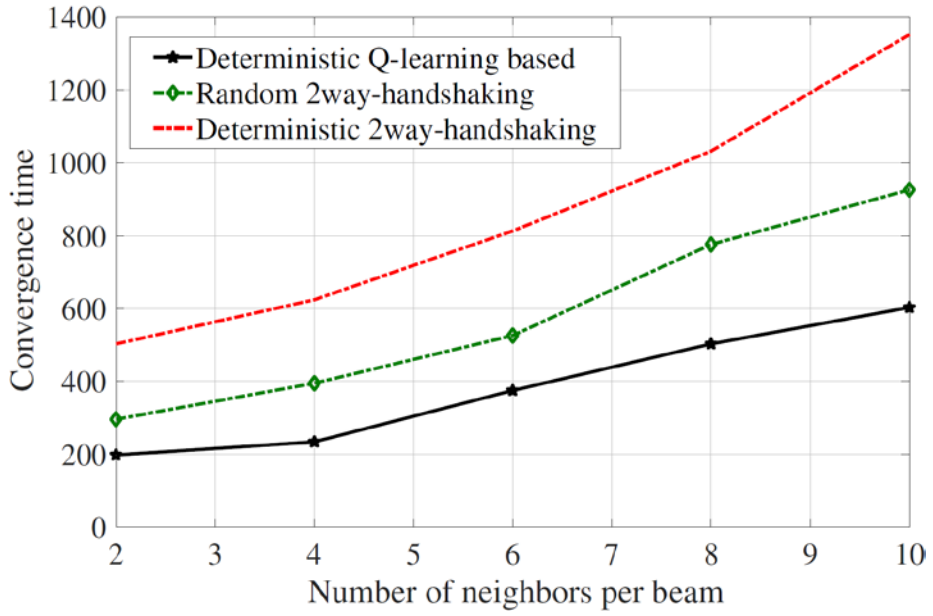


Figure 30. Average convergence time for $k = 6$ and varying node density.

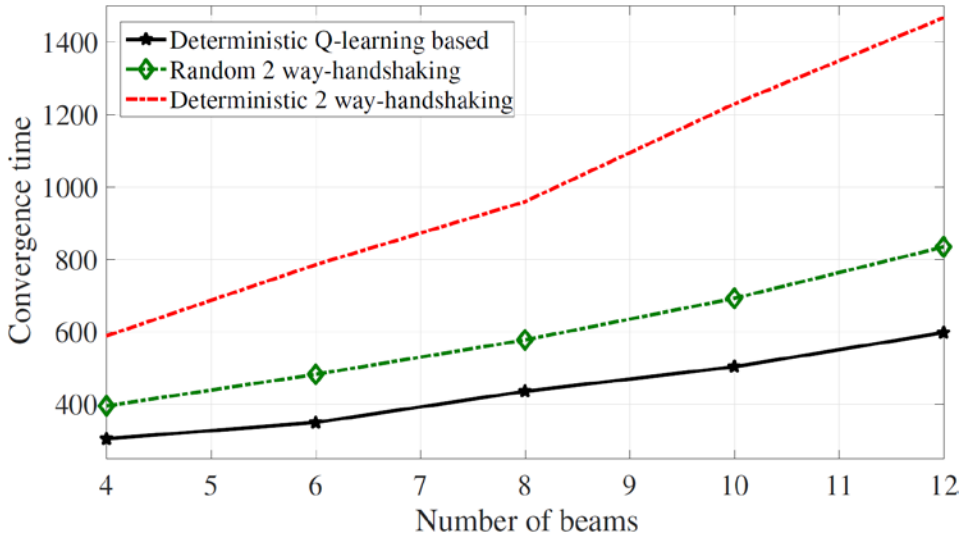


Figure 31. Average convergence time for $Nei = 5$ and different beamwidths.

### 4.3.4. Discussions

The value of the exploration parameter $\varepsilon$ reflects the need to explore more alternative actions in a given state, as the value of optimal action in both the deterministic and probabilistic algorithms is different in each time slot. Intuitively, the value of $\varepsilon$ depends on the network density. For instance, for a network with high density (more than three nodes per beam), exploring new state-action pairs helps in discovering multiple neighbors in the same beam with minimal latency. On the other hand, a small value of $\varepsilon$ becomes more appropriate for low density networks as a greedy approach guarantees a faster discovery [22].

The same logic can be used for learning parameter $\beta$. As dense networks are more prone to collisions, a high value of $\beta$ will allow the colliding nodes to discover each other in a relatively short period of time by giving high Q-values to the state-action pairs experiencing collisions. In contrast, the sparse networks need a smaller value of the learning parameter to allow the exploitation of information collected by investigating other sectors. A good estimate of the step size for $\beta$ and $\varepsilon$ can be found through numerical simulations as suggested in [26].

# 5. CONCLUSIONS

Use of purely directional antennas in wireless mesh networks have many advantages, including increased transmisison range, energy conservation, higher throughout due to the spatial reuse and interference avoidance. However, the network consisting of directional antennas (without the support of omni-directional antennas) requires a significant neighbor discovery overhead and large latency because a node can communicate with its neighbors in only a narrow area at a time due to the limited antenna beamwidth. In this project, we designed three novel directional neighbor discovery schemes which have significantly lower protocol overhead and discovery latency. These schemes intelligently consider the collisions and use machine learning techniques. We have also built a hardware testbed to test the performance of neighbor discovery schemes for WMN.

We first designed a collision-aware neighbor discovery algorithms for synchronous static ad-hoc networks with directional antenna. By introducing the collision resolving mechanism, we addressed the lack of collision management in the earlier approaches. We also extended our scheme by enabling cooperation between nodes. As expected the peer assisted collision-aware scheme performs favorably compared to scan based and random neighbor discovery schemes. To prove the effectiveness of our scheme, we assessed its improvement in a variety of networks with increasing networks density and different number of sectors. Numerical results show that our proposed scheme performs well in the collision-prone scenarios (wide beams and/or high node density).

Next, we designed a learning automata based algorithm to expedite the neighbor discovery process in directional wireless networks, which allows a node to learn from success and collision history. The simulation results demonstrated that the use of learning algorithm exhibits a significant performance improvement over existing random and scan based neighbor discovery schemes for different node densities and beamwidths. Our proposed scheme performs particularly well in networks with narrow beamwidth and high node density.

Finally, we designed a new distributed Q-learning based scheme that allow a directional node to learn from the successful neighbor discovery and collision history to improve the neighbor discovery rate. This deterministic algorithm help the nodes to decide the most appropriate operating mode in order to achieve a rapid convergence. The numerical results demonstrate that the our scheme exhibits a significant improvement over the two-way handshaking random and deterministic schemes for different scenarios. Furthermore, the designed schemes perform particularly well in networks with high density.

Hardware-based demos are very important for practical protocol evaluation. We used the USRP-RIO boards to establish a directional network platform. In this DND testbed, we demonstrated the neighbor discovery process through two-beam directional antennas. We first achieved a basic neighbor discovery function in Demo 1, where the USRP RF board useed one or two beams to search valid neighbors in different directions. In Demo 2, we further explored the capabilities of USRP boards in terms of concurrent neighbor discovery in different beam directions. Two directional antennas are used to search valid neighbors simultaneously. Additionally, the video streams were also transmitted and played in real-time via the two antennas of the USRP board at the same time. In Demo 3, we used the concept of signal quality sensing during the DND process and used it for the transmission performance optimization.

# 6. REFERENCES


[1] DirecNet Waveform Overview and Architecture Document, May 2014.

[2] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "Using directional antennas for medium access control in ad hoc networks," In *Proc. 8th Annual Intl Conf. Mobile Computing and Networking*, MobiCom, pages 59–70, New York, NY, 2002.

[3] G. Jakllari, W. Luo, and S. V. Krishnamurthy, "An integrated neighbor discovery and MAC protocol for ad hoc networks using directional antennas," *IEEE Trans. Wireless Commun.*, 6(3):1114–1124, Mar. 2007.

[4] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, and S. Polit, "Ad hoc networking with directional antennas: A complete system solution," *IEEE J. Select. Areas in Commun.*, 23(3):496–506, Mar. 2005.

[5] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley, "Efficient algorithms for neighbor discovery in wireless networks," *IEEE/ACM Trans. Networking*, 21(1):69–83, Feb. 2013.

[6] S. Vasudevan, J. Kurose, and D. Towsley. "On neighbor discovery in wireless networks with directional antennas," *Proc. IEEE 24th Annual Joint Conf. of Computer and Communications Societies*, volume 4, pages 2502–2512, Mar. 2005.

[7] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrman, and A. Manjeshwar, "Energy-efficient link assessment in wireless sensor networks," In *IEEE INFOCOM*, Hong Kong, 2004.

[8] M. McGlynn and S. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in wireless networks," *Proc. ACM MOBIHOC*, Long Beach, CA, 2001.

[9] NI USRP-RIO node: see http://sine.ni.com/nips/cds/view/p/lang/en/nid/212991.

[10] Z. H. Mir, W. S. Jung, and Y. B. Ko, "Continuous Neighbor Discovery Protocol in Wireless Ad Hoc Networks with Sectored-Antennas, " In *29th IEEEInternational Advanced Information Networking and Applications*, pages 54–61, March 2015.

[11] R. Nichols, "Directional network discovery performance," In *34th IEEE Sarnoff Symposium*, pages 1–5, May 2011.

[12] Z. Zhang and B. Li, "Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons," *IEEE Trans. Wireless Commun.*, vol. 7, no. 5, pp. 1540–1549, May 2008.

[13] H. Cai and T. Wolf, "On 2-way neighbor discovery in wireless networks with directional antennas," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2015, pp. 702–710.

[14] S. Zhao, Y. Liu, T. Yang, Z. Feng, Q. Zhang, and C. Gao, "3-way multi-carrier asynchronous neighbor discovery algorithm using directional antennas," in *Proc. IEEE Wireless Commun. and Networking Conf.*, Apr. 2016, pp. 1–6.

[15] S. Vasudevan, D. Towsley, D. Goeckel, and R. Khalili, "Neighbor discovery in wireless networks and the coupon collector's problem," in *Proc. Int. Conf. Mobile Computing and*

*Networking*, 2009, pp. 181–192

[16] B.El khamlichi, D. H. N. Nguyen, J. El Abbadi, N. W. Rowe, and S. Kumar,"Collision-Aware Neighbor Discovery with Directional Antennas, in *Int. Conf. Computing, Networking and Communications*, March 2018, Hawaii, USA.

[17] L. Chen, Y. Li, and A. V. Vasilakos, "Oblivious neighbor discovery for wireless devices with directional antennas," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[18] X. An, R. V. Prasad, and I. Niemegeers, "Impact of antenna pattern and link model on directional neighbor discovery in 60 GHz networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 5, pp. 1435–1447, May 2011.

[19] G. M. Olser, Z. Genc, and E. Onur, "Smart neighbor scanning with directional antennas in 60 GHz indoor networks," in *Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun.*, 2010, pp. 2393–2398.

[20] 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 *Proc. Annual Conf. on Sensor, Mesh and Ad Hoc Commun. and Networks*, Jun. 2010, pp. 1–9.

[21] E. Gelal, G. Jakllari, S. V. Krishnamurthy, and N. E. Young, "Topology management in directional antenna-equipped ad hoc networks," *IEEE Trans. Mobile Computing*, vol. 8, pp. 590â€"605, May 2009.

[22] B. E. Khamlichi, D. H. N. Nguyen, J. E. Abbadi, N. W. Rowe, and S. Kumar, Learning Automaton Based Neighbor Discovery for Wireless Networks Using Directional Antennas, to appear in IEEE Wireless Communications Letters.

[23] F. Tian, B. Liu, H. Cai, H. Zhou, and L. Gui, "Practical Asynchronous Neighbor Discovery in Ad Hoc Networks With Directional Antennas," *IEEE Trans. Vehicular Technology*, 65(5):3614–3627, May 2016.

[24] S. Vasudevan, D. Towsley, D. Goeckel, and R. Khalili, "Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem," In *Proc. 15th ACM Annual International Mobile Computing and Networking* (MobiCom '09), pages 181–192, 2009.

[25] B. J. Oommen and M. A. L. Thathachar, "Multi-action learning automata possessing ergodicity of the mean," *Information Sciences*, vol. 35, pp. 183–198, June 1985.

[26] R. Cohen and B. Kapchits, "A Continuous Neighbor Discovery in Asynchronous Sensor Networks," *IEEE Trans. Networking*, vol. 19, pp. 69-79, Feb. 2011.

[27] C. J. C. H. Watkins, "Learning From Delayed Rewards", Ph.D dissertation, Cambridge Univ., U.K., 1989.

[28] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, May 1992.

[29] M. Wunder, M. Littman, and M. Babes-Vroman, "Classes of multiagent Q-learning dynamics with e-greedy exploration," *Int. Conf. Machine Learning*, 2010, pp. 1167–1174.

[30] S. B. Thrun, Coefficient exploration in reinforcement learning, *Technical Report*, 1992.

[31] W. Xiong, B. Liu, and L. Gui, "Neighbor Discovery with Directional Antennas in Mobile Ad-Hoc Networks," In *GLOBECOM 2011*, pages 1–5, December 2011.

# 7. LIST OF ABBREVIATIONS, AND ACRONYMS

| | |
|---|---|
| ACK | Acknowledgement |
| ADC | Analog to Digital Converter |
| BER | Bit Error Rate |
| CBR | Constant Bit Rate |
| CSMA | Carrier Sense Multiple Access |
| CW | Contention Window |
| DAC | Digital to Analog Converter |
| DND | Directional Neighbor Discovery |
| FLA | Finite-state Learning Algorithm |
| FPGA | Field Programmable Gateway Access |
| GPS | Global Positioning |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| LA | Learning Automaton |
| MAC | Medium Access Control |
| MARL | Multi Agent Reinforcement Learning |
| MIMO | Multi-Input Multi-Output |
| NCAA | National Collegiate Athletic Association |
| PDR | Packer Dropping Rate |
| PER | Packet Error Rate |
| QoS | Quality of Service |
| RF | Radio Frequency |
| RL | Reinforced Learning |
| Rx | Receiver |
| SBA | Scan-based Algorithm |
| SBA-D | Deterministic Scan Based Algorithm |
| SBA-R | Random Scan Based Algorithm |
| SNRs | Signal-to-Noise Ratios |
| Tx | Transmitter |
| USRP | Universal Software Radio Peripheral |
| WMN | Wireless Mesh Networks |