



On neighbor discovery in cognitive radio networks

Neeraj Mittal^{*}, Srinivasan Krishnamurthy, R. Chandrasekaran, S. Venkatesan, Yanyan Zeng

Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA

ARTICLE INFO

Article history:

Received 3 February 2008

Received in revised form

29 January 2009

Accepted 23 March 2009

Available online 8 April 2009

Keywords:

Cognitive radio network

Multiple channels

Neighbor discovery

Adaptive algorithm

Termination detection

Collision-awareness

ABSTRACT

A cognitive radio node is a radio device capable of operating over multiple channels. As a result, a network consisting of one or more cognitive radio nodes can *adapt* to varying channel availability in its geographical region by dynamically changing the channel (or channels) nodes are using for communication.

We investigate the problem of designing a *fast* deterministic algorithm for *neighbor discovery* in an undirected network consisting of one or more cognitive radio nodes when different nodes may have different subsets of channels available for communication at their location. We say that a neighbor discovery algorithm is fast if its time-complexity is polynomial in actual network-size. We prove that it is impossible to devise a fast algorithm to solve the neighbor discovery problem if nodes are neither *size-aware* nor *collision-aware*.

When nodes are collision-aware, we present a fast algorithm for neighbor discovery with time-complexity of $O(p m \log n)$, where p denotes the actual number of nodes present in the network, n denotes the size of space used for assigning labels to the nodes, and m denotes the number of channels over which nodes can operate. Our algorithm has the desirable property that every node knows when the neighbor discovery has completed and, moreover, all nodes terminate at the same time. When nodes are size-aware, we use a gossiping algorithm for a single-channel network with large labels that was proposed recently to derive a fast adaptive algorithm for neighbor discovery. We also investigate the neighbor discovery problem when the network may be directed. Finally, we describe a way to speed up neighbor discovery when nodes are collision-aware and channel availability sets of neighboring nodes do not vary significantly.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

With the unprecedented proliferation of wireless communication devices in recent times, it is generally believed that there will be an acute shortage of bandwidth in the near future. This belief is supported by the observation that most of the available frequency bands have already been allocated to various applications (e.g. television transmission, microwave communication, cellular communication, etc.) [8]. However, it has been noted that a significant portion of the allocated spectrum is under-utilized [9,3]. For example, in several rural areas many of the television channels in the VHF and UHF bands are unassigned.

Cognitive Radio (CR) technology [18] allows wireless devices to dynamically adapt to spectrum availability in their geographical region. The owner of a licensed channel is referred to as *primary user* and all other users of the channel as *secondary users* [3]. CR technology enables secondary users to scan and identify unused channels in a frequency spectrum. A channel is said to be *available*

if the secondary user can send and receive messages on the channel without interfering with the primary user(s). A communication infrastructure based on CR technology has applications in defense and relief & rescue operations. Since the spectrum usage varies widely from one region to another [9], communication among users (soldiers in a platoon or relief personnel in a disaster-prone area) must rely on a dynamic channel assignment scheme. When a secondary user (hereafter, referred to as a node in the CR network) independently scans the spectrum usage and maintains the set of locally available channels, which may be different for different nodes, the following layer-2 auto-configuration issue arises [2]: How do nodes detect their neighbors and collectively form a communication infrastructure in the absence of a central authority? This gives rise to the *neighbor discovery problem*, also referred to as the *neighbor gossip problem* in the literature. We say that two nodes are neighbors if they are within wireless transmission range of each other.

Neighbor discovery is an important problem in radio networks (especially, static radio networks) because it can be used as a primitive for solving other important communication problems, such as broadcasting a message [4,5,13,14,7,12,15], finding a globally common channel [16] or computing a deterministic

^{*} Corresponding author.

E-mail address: neerajm@utdallas.edu (N. Mittal).

transmission schedule [21], in a more efficient manner. For instance, if each node in the network knows its neighbors, then broadcasting and gossiping problems can be solved in almost linear time using almost linear time leader election and linear time DFS traversal [11]. Clearly, it is desirable to first perform neighbor discovery if several messages have to be broadcast one after another and the neighborhood itself is not expected to change significantly during this duration.

In this paper, we investigate the neighbor discovery problem under the following system model: There is a synchronous radio network consisting of p cognitive radio nodes capable of operating over m channels. Different nodes may have different subsets of channels available for communication at their location at any given time. Time is slotted and all nodes run a deterministic algorithm. Each node is assigned a unique label selected from the range $[0, n - 1]$. All nodes know n and m but not p . For convenience, we assume that n is a power of two. Unless otherwise stated, the network is assumed to be undirected; that is, all links are bidirectional. We are interested in designing a fast neighbor discovery algorithm whose time-complexity is polynomial in actual network-size and only poly-logarithmic in maximum network-size; that is, its time-complexity is $O(\text{poly}(p)\text{polylog}(n) m)$.

Extensive research has been conducted over the last few decades on devising *deterministic* solutions to several important communication problems in *single-channel* ad hoc radio networks, including broadcasting [4,13,14,7,22,12,15], gossiping [5,11,22] and global function computation [10], under a variety of assumptions. The gossiping problem is a natural extension of the broadcasting problem in which every node has a message to send to all other nodes in the network [5,11,22]. The global function computation is a generalization of the gossip problem in which every node holds a value and some function (e.g., maximum or average) has to be computed using the values held by all of the nodes [10]. Some of the algorithms that have been proposed even solve the neighbor discovery problem as a part of solving the original problem (e.g., [4,22]). However, all the algorithms for solving these problems have been developed under a system model that is different from the system model considered in this paper. Specifically, the system model typically assumed in the literature is that nodes know neither p nor n but n is assumed to be $O(p)$ [4,5,13,14,7,12,15]. Further, the entire network operates on the *same channel*. In most cases, although the algorithm itself is guaranteed to terminate within $O(\text{poly}(p))$ time-slots, no node may know that the algorithm has actually terminated. Chlebus et al. [4], in fact, show that termination of a broadcasting algorithm cannot be detected in general unless nodes can detect collisions. Uchida et al. [22] show that it is possible to solve the terminating versions of broadcasting and gossiping problems even if nodes cannot detect collisions as long as the network contains at least two nodes.

In our system model, nodes know n but not p , and no relationship is assumed to exist between p and n except that p is at most n . In particular, p may be much smaller than n , or, in other words, the label space size may be much larger than the number of nodes in the network. For example, if nodes are assigned 32-bit identifiers, then label space size is 2^{32} but the network may contain only hundreds of nodes. Furthermore, in our model, nodes can operate over multiple channels. This introduces an additional level of complexity. For two neighboring nodes to communicate successfully, both of them should be tuned to the *same channel* at the *same time* in addition to the requirement that the sending node's transmission should not collide with another node's transmission which is also within the communication range of the receiving node. Since different nodes may have different sets of channels available at their respective locations, nodes may have to perform neighbor discovery on more than one channel in their availability sets. Therefore it is important for all nodes

to be able to detect the completion of neighbor discovery on one channel *simultaneously* and switch to the next channel in a *coordinated* manner. This is especially hard to achieve in an *adaptive* algorithm in which a step of a node may depend on the knowledge it has gained so far because the sub-network induced on one channel may be different from the sub-network induced on another channel. Moreover, even if the network as a whole is connected, the sub-network induced on a channel may not be, and may consist of several connected components of vastly different sizes. As a result, different nodes may have quite different views of the network when restricted to their respective channel availability sets. Note that, in this paper, we use the word "adaptive" in two different ways. A cognitive radio network is adaptive because it can operate over multiple channels. An algorithm is adaptive because next step of a process may depend on the knowledge the process has gained so far.

We make the following contributions in this paper. First, we prove that it is impossible to devise a fast algorithm to solve the neighbor discovery problem if nodes are neither *size-aware* (that is, they know a polynomial upper-bound on the actual number of nodes present in the network) nor *collision-aware* (that is, they can distinguish between background noise and collision noise). This lower-bound result can be seen as an extension of the lower-bound result in [16], which was proved under the assumption that the neighbor discovery algorithm is collision free. Second, we present a fast adaptive algorithm for neighbor discovery when nodes are collision-aware (but not size-aware). Our algorithm has time-complexity of $O(pm \log n)$, and has the desirable property that every node knows when the neighbor discovery has completed. Moreover, all nodes terminate at the same time. For the sake of completeness, we use the gossiping algorithm for large labels proposed by Gąsieniec et al. [11] to obtain a fast algorithm for neighbor discovery when nodes are size-aware. Third, we present a fast neighbor discovery algorithm when the network may be directed (that is, some links may be unidirectional) and nodes are collision-aware. The algorithm has time-complexity of $O(p D m \log n)$, where D is the diameter of the network. Finally, when nodes are collision-aware, we show that the time-complexity of our neighbor discovery algorithms can be reduced if there is a known bound b on the divergence in the channel availability sets of neighboring nodes and $b < \sqrt{m}$.

In many ad hoc radio networks, no explicit neighbor discovery algorithm is used [17]. Rather, nodes transmit heartbeat messages periodically; nodes discover their neighbors by constantly listening for these heartbeat messages when not transmitting (e.g., [20]). This approach, besides being simple, is quite suitable when nodes are mobile and neighborhood relationships may not last for a long time. However, when nodes are relatively static, it is more efficient to explicitly discover neighborhood relationships [17]. Further, the performance of the approach based on periodic heartbeat messages degrades progressively as the network becomes more dense due to increased collisions in heartbeat messages.

Neighbor discovery algorithms in [1,16], which use a round-robin schedule, have high asymptotic time-complexity that is linear in n and are therefore slow according to our classification. The neighbor discovery algorithms proposed by Vasudevan et al. in [23] assume a single-channel network in which nodes have *directional* antennas. Borbash et al. describe a stochastic neighbor discovery algorithm for a single-channel asynchronous wireless sensor network in [2]. Their algorithm, however, requires every node to know a good estimate of the size of its neighborhood *a priori*. The performance of the algorithm degrades if the estimate is inaccurate. Zheng et al. propose a neighbor discovery algorithm in [24] under the assumption that nodes can simultaneously send and receive on the same channel, which is difficult to achieve in practice.

Table 1

Comparison of various neighbor discovery algorithms for cognitive radio networks.

Assumptions				Time-complexity	Reference
Network is undirected	Nodes are collision-aware	Nodes are size-aware	Channel divergence is bounded		
Yes	No	No	No	nm	[16]
Yes	No	No	Yes	$n(b+1)^2$	[16]
Yes	Yes	No	No	$O(pm \log n)$	[this paper]
Yes	Yes	No	Yes	$O(p(b+1)^2 \log n)$	[this paper]
Yes	No	Yes	No	$O(pm \log^2 p \log^2 n)$	[this paper]
No	Yes	No	No	$O(pDm \log n)$	[this paper]
No	Yes	No	Yes	$O(pD(b+1)^2 \log n)$	[this paper]

p : number of nodes in the network; n : size of space used for assigning unique labels to nodes; m : number of channels over which nodes can operate; b : bound on divergence in channel availability sets of two neighboring nodes; D : diameter of the network.

Our work in this paper is, therefore, different from the existing work in many ways. First, we investigate the neighbor discovery problem under a different system model: nodes know n and m but not p , no relationship is assumed to exist between p and n , and $m \geq 1$. Second, nodes can not only operate over multiple channels but different nodes may have different channel availability sets. Third, the neighbor discovery algorithm proposed in this paper provides strong termination guarantees. This is a desirable property to have in a multi-channel radio network if the results of neighbor discovery are to be used subsequently to perform other more complex communication tasks. Fourth, nodes may not have directional antennas. We are not aware of any prior work on designing a fast deterministic neighbor discovery algorithm in a multi-channel network with termination detection capability for the above-described system model. The only other work somewhat related to our work is by Gąsieniec et al. [11], in which the gossiping problem is solved when the label space is large and known to all nodes but all nodes operate on the same channel and no node can detect termination of the gossiping algorithm. Alternatively, strong and simultaneous termination can be achieved if the nodes are size-aware. Observe that we are able to provide strong termination guarantees because we assume a stronger model in which nodes can detect collisions. Gąsieniec et al. [11], on the other hand, assume a weaker model in which nodes cannot detect collisions. Table 1 compares various neighbor discovery algorithms for multi-channel networks that have been proposed so far to our knowledge, including the ones proposed in this paper.

To summarize, we make the following contributions in this paper:

- We show that it is impossible to devise a fast neighbor discovery algorithm for an undirected network unless nodes are either collision-aware or size-aware.
- We present fast neighbor discovery algorithms when the network is undirected and nodes are either collision-aware or size-aware.
- We present a fast neighbor discovery algorithm when the network is directed and nodes are collision-aware.
- We describe a way to speed up neighbor discovery when nodes are collision-aware and channel availability sets of neighboring nodes do not differ significantly.

The rest of the paper is organized as follows. We describe the system model formally in Section 2, and prove the lower-bound in Section 3. We present fast neighbor discovery algorithms when nodes are collision-aware in Section 4. We present a fast neighbor discovery algorithm when nodes are size-aware, which is based on the gossiping algorithm by Gąsieniec et al. [11] for large labels, in Section 5. We investigate the neighbor discovery problem for directed networks in Section 6. Finally, we present our conclusions and outline directions for future research in Section 7.

2. System model

Unless otherwise indicated, we assume a multi-hop bidirectional (or symmetric) wireless network consisting of one or more cognitive radio nodes, referred to as a *cognitive radio network*. Each cognitive radio node has a single transceiver (that is, transmitter–receiver pair), which is capable of operating over multiple channels. However, at any given time, a transceiver can operate (either transmit or receive) only over a single channel. Further, a transceiver cannot transmit and receive at the same time.

We assume that the network is homogeneous in the sense that all radio nodes are capable of operating over the same set of channels, referred to as the *universal channel set*. However, the set of channels available at a node may be different at different nodes. The exact set of channels available at a node depends on the channel usage by other nodes (external to the network) present in the vicinity of that node. We use $\mathcal{UCS} = \{c_1, c_2, \dots, c_m\}$ to denote the universal set of channels.

We assume that each cognitive radio node is assigned a *unique label* selected from the range $[0, n-1]$. This in turn implies that the network can contain at most n nodes. Hereafter, unless otherwise stated, assume that n is a power of two. For a node r , we use $\text{label}(r)$ to refer to the unique identifier assigned to r . Note that the size of a label is logarithmic in n . The actual number of nodes present in the network is denoted by p . Hereafter, we use the terms “node” and “cognitive radio node” interchangeably.

We say that a node is *size-aware* if it knows a polynomial upper-bound on the actual number of nodes present in the network; otherwise, it is *size-unaware*. Further, we say that a node is *collision-aware* if it can distinguish between background noise and noise due to a collision; otherwise, it is *collision-unaware*. Hereafter, unless otherwise stated, we assume that nodes are size-unaware. Later, we study the neighbor discovery problem when nodes are size-aware.

We assume that an execution of the system is divided into *time-slots*. In each time slot, a node can be in one of the following three modes: (1) *transmit mode* on some channel in its availability set, (2) *receive mode* on some channel in its availability set, or (3) *quiet mode* when the transceiver is shut-off.

For convenience, we assume that the network is “connected” in the sense that there is a path between every pair of nodes where different edges in the path may be operating on different channels. Otherwise, neighbor discovery is simply performed in each “connected” component of the network. Note that the sub-network induced on any one channel may not be connected.

2.1. The neighbor discovery problem

We assume that, when the system starts, different nodes do not know about each other; that is, each node only possesses knowledge about itself and what is common knowledge among nodes. Specifically, a node r initially knows about: (1) the universal

channel set: \mathcal{UC}_s , (2) the size of the \mathcal{UC}_s : m , (3) the range from which labels have been chosen: $[0, n - 1]$, (4) the unique label assigned to r : $\text{label}(r)$, and (5) the subset of channels available at r .

To learn about their respective neighbors, nodes execute a special algorithm referred to as the neighbor discovery algorithm. Neighbor discovery is said to have finished *locally* at a node if the node has learned about all its neighbors and all its neighbors have learned about it. Neighbor discovery is said to have finished *globally* if it has finished locally at every node in the network. Any neighbor discovery algorithm should satisfy the following properties:

- **No false discovery:** a node considers another node to be its neighbor only if the two nodes are neighbors of each other.
- **Eventual discovery:** every node eventually discovers all its neighbors.
- **Termination:** neighbor discovery at every node eventually terminates.

The termination condition can be defined in several ways. A node is said to have *weakly terminated* if it knows that the neighbor discovery algorithm has finished locally at the node. A node is said to have *strongly terminated* if it knows that the neighbor discovery algorithm has finished globally. Note that it is preferable to use the weaker definition of termination for the lower-bound proof and the stronger definition of termination for the actual algorithm.

In this paper, we focus on designing a deterministic neighbor discovery algorithm that is *fast*; that is, its time-complexity is polynomial in actual network-size (which is given by p) and only poly-logarithmic in maximum network-size (which is given by n).

3. Lower-bound in the absence of collision awareness

We now show that it is *impossible* to devise a fast deterministic neighbor discovery algorithm even under a *weak* termination condition if nodes are neither size-aware nor collision-aware. First, we explain the main idea behind our lower-bound proof. Next, we define some concepts and notations needed to describe and understand the proof. Finally, we present the lower-bound proof.

3.1. The main idea

The lower-bound on the time-complexity of neighbor discovery is based on the notion of exclusive channel set. The exclusive channel set of a node in the network is the subset of channels in its availability set that are “exclusive” to that node in the sense that the node does not share any of those channels with any of its neighbors. We show that neighbor discovery for a node cannot be complete until the node has tuned its receiver to each channel in its exclusive set *at least once* for each node *not present* in the network. Otherwise, we can construct two different network configurations such that: (1) the node has different sets of neighbors in the two configurations, and (2) the node cannot distinguish between the two configurations. This violates the *correctness* of the neighbor discovery algorithm.

Let x denote the maximum size of an exclusive channel set among all exclusive channel sets in the network. It follows from the above-mentioned result that any deterministic algorithm for neighbor discovery has to use at least $(n - p) \times x$ time-slots before it can terminate. Clearly, for any configuration for which $p = o(n)$, $n - p = \Theta(n)$. As a result, the time-complexity of any deterministic neighbor discovery algorithm in the worst case is $\Omega(nm)$ whenever x is $\Theta(m)$ even when p is much smaller than n . We now describe some concepts and notations used in our lower-bound proof.

3.2. Network configuration

We model a cognitive radio network using a *configuration*. A configuration \mathcal{C} is a 3-tuple $\langle \mathcal{P}, \mathcal{N}, \mathcal{A} \rangle$, where:

- \mathcal{P} denotes the set of nodes present in the network,
- \mathcal{N} denotes the neighborhood function that maps each node in the network to a subset of nodes in the network; that is,

$$\mathcal{N} : \mathcal{P} \longrightarrow 2^{\mathcal{P}},$$
- \mathcal{A} denotes the channel availability function that maps each node in the network to a non-empty subset of channels of the universal channel set; that is,

$$\mathcal{A} : \mathcal{P} \longrightarrow 2^{\mathcal{UC}_s} \setminus \emptyset.$$

Intuitively, the neighborhood function captures the set of nodes that are within communication range of each node in the network. Further, the channel availability function captures the set of channels that are currently available for use at each node in the network. We assume that the neighborhood function is *symmetric*. Specifically, for two nodes r and s in \mathcal{P} :

$$s \in \mathcal{N}(r) \iff r \in \mathcal{N}(s).$$

Observe that, even if two nodes are within communication range of each other, they can communicate only if their channel availability sets contain at least one common channel. Therefore, for a node r , we define the *effective* neighborhood function, denoted by $\mathcal{N}_e(r)$, as follows:

$$\mathcal{N}_e(r) \triangleq \{s | s \in \mathcal{N}(r) \text{ and } \mathcal{A}(r) \cap \mathcal{A}(s) \neq \emptyset\}.$$

Observe that the effective neighborhood function is symmetric if the neighborhood function is symmetric. We assume that the communication graph induced by the effective neighborhood function is *connected*. Hereafter, unless otherwise stated, we use the term “neighbor” to refer to an “effective neighbor”.

3.3. Network execution

A *state* of a node captures the knowledge that the node has gained about the network so far. Further, a node can be in any of the three modes (transmit, receive, or quiet) during a time-slot. A *state assignment* (respectively, *mode assignment*) is an assignment of state (respectively, mode) to every node in the network. Intuitively, a network execution is an alternating sequence of state assignment and mode assignment, starting with the initial state assignment. For a node r , execution α and time-slot t , we use $\text{state}(r, \alpha, t)$ to denote the state of node r at the end of time-slot t in execution α . Likewise, we use $\text{mode}(r, \alpha, t)$ to denote the mode (transmit, receive or quiet) of node r during time-slot t in execution α . Note that $\text{state}(r, \alpha, 0)$ denotes the *initial* state of node r in execution α . In this paper, we focus on deterministic algorithms in which the mode of a node during a time-slot is only a function of its state in the previous time-slot.

When a node is listening on some channel during a time-slot, then either of the following two possibilities can occur: (1) the node hears a clear message, or (2) the node does not hear anything. We define the *view* of a node during a time-slot as what it hears during the time-slot in case its mode is to receive (on some channel in its availability set) during that time-slot and \perp otherwise. For a node r , execution α and time-slot t , we use $\text{view}(r, \alpha, t)$ to denote the view of node r during time-slot t in execution α . Note that, in a deterministic algorithm, the state of a node at the end of a time-slot depends only on its state at the beginning of the time-slot and its view during the time-slot.

We show in [16] that any deterministic neighbor discovery algorithm has a *unique* execution for every network configuration.

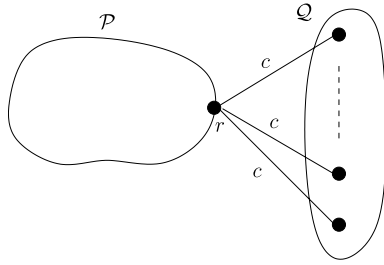


Fig. 1. A derived configuration for a node r and a channel c in its exclusive channel set.

Theorem 1 (*Determinism \Rightarrow Uniqueness [16]*). Any deterministic algorithm for neighbor discovery has a unique execution for every network configuration.

For an execution α and time-slot t , we use $\alpha|t$ to denote the prefix of the execution α up to and including the time-slot t .

3.4. Indistinguishable executions

Central to our lower-bound proof is the notion of *indistinguishable executions*. Intuitively, two executions are said to be indistinguishable by a node until a time-slot if it has the same initial state and the same sequence of views up to and including that time-slot in both executions. Formally, two executions α and β are indistinguishable by node r up to time-slot t , denoted by $\alpha \xleftrightarrow[r]{t} \beta$, if the following two conditions hold:

- the initial state of node r in execution α is same as that in execution β ; that is, $state(r, \alpha, 0) = state(r, \beta, 0)$,
- the sequence of views of node r in execution α is same as that in execution β at least until time-slot t ; that is, $(\forall u : u \leq t : view(r, \alpha, u) = view(r, \beta, u))$.

3.5. The lower-bound proof

Consider a configuration $\mathcal{C} = \langle \mathcal{P}, \mathcal{N}, \mathcal{A} \rangle$. For a node r in the network, its *exclusive channel set*, denoted by $\mathcal{X}(r)$, is defined as

$$\mathcal{X}(r) \triangleq \{c \in \mathcal{A}(r) | \forall s : s \in \mathcal{N}(r) : c \notin \mathcal{A}(s)\}.$$

Now, consider a channel c in the exclusive channel set $\mathcal{X}(r)$. Let \mathcal{Q} be a subset of nodes not present in the network. (More precisely, \mathcal{Q} is a subset of labels from the range $[0, n - 1]$ that are “missing” from the network.) We construct another configuration derived from the base configuration \mathcal{C} in which all nodes in \mathcal{Q} are neighbors of node r (and only node r) on channel c (see Fig. 1). Formally, the derived configuration, denoted by $\mathcal{D}(r, c, \mathcal{Q})$, is given by the 3-tuple $\langle \mathcal{P} \cup \mathcal{Q}, \mathcal{M}, \mathcal{B} \rangle$, with \mathcal{M} defined as

$$\mathcal{M}(s) \triangleq \begin{cases} \mathcal{N}(s) : s \in \mathcal{P} \setminus \{r\} \\ \mathcal{N}(s) \cup \mathcal{Q} : s = r \\ \{r\} : s \in \mathcal{Q} \end{cases}$$

and \mathcal{B} defined as

$$\mathcal{B}(s) \triangleq \begin{cases} \mathcal{A}(s) : s \in \mathcal{P} \\ \{c\} : s \in \mathcal{Q}. \end{cases}$$

Fix a deterministic neighbor discovery algorithm NDA. We use α to denote the unique execution of the algorithm NDA for the configuration \mathcal{C} . Further, we use $\beta(r, c, \mathcal{Q})$ to denote the unique

execution of the algorithm NDA for the derived configuration $\mathcal{D}(r, c, \mathcal{Q})$.

Let \mathcal{Q}_0 denote the set of *all* nodes not present in the network. Let $w = n - p$. Note that $p = |\mathcal{P}|$ and $w = |\mathcal{Q}_0|$. We show that there exists a sequence of subsets of nodes not present in the network, given by $\mathcal{Q}_0, \mathcal{Q}_1, \dots, \mathcal{Q}_w$, and a sequence of time-slots, given by t_0, t_1, \dots, t_w , such that: (P1) node r cannot distinguish between executions α and $\beta(r, c, \mathcal{Q}_i)$ up to time-slot t_i , and (P2) the prefix of the execution α up to and including time-slot t_i contains at least i time-slots during which node r is in receive mode on channel c .

We construct the two sequences inductively. The two properties clearly hold for \mathcal{Q}_0 and t_0 if t_0 is defined to be 0. We now show how to inductively compute \mathcal{Q}_{i+1} and t_{i+1} given \mathcal{Q}_i and t_i with $i < w$ satisfying the two properties. During our construction, we maintain the following invariants: (I1) $\mathcal{Q}_{i+1} \subseteq \mathcal{Q}_i$ and $t_{i+1} > t_i$ for each $i < w$, and (I2) $|\mathcal{Q}_i| \geq w - i$ for each $i \leq w$. Note that the second invariant trivially holds for \mathcal{Q}_0 . Further, it follows from the second invariant that, whenever $i < w$, $|\mathcal{Q}_i| \geq 1$. This implies that node r has at least one additional neighbor in the configuration $\mathcal{D}(r, c, \mathcal{Q}_i)$ when compared to the configuration \mathcal{C} whenever $i < w$. In other words, the two configurations are *different* with respect to node r .

For ease of exposition only, we fix node r and channel c , and use $\mathcal{D}(\mathcal{Q}_i)$ and $\beta(\mathcal{Q}_i)$ to refer to $\mathcal{D}(r, c, \mathcal{Q}_i)$ and $\beta(r, c, \mathcal{Q}_i)$, respectively. Consider executions α and $\beta(\mathcal{Q}_i)$ for configurations \mathcal{C} and $\mathcal{D}(\mathcal{Q}_i)$, respectively, where $i < w$. By the two properties (P1) and (P2), node r cannot distinguish between execution α and $\beta(\mathcal{Q}_i)$ up to time-slot t_i . However, as argued before, configurations \mathcal{C} and $\mathcal{D}(\mathcal{Q}_i)$ are different with respect to node r . Note that node r has not yet weakly terminated in $\alpha|t_i$. Otherwise, it implies that node r has weakly terminated in $\beta(\mathcal{Q}_i)|t_i$ as well, thereby violating the correctness of neighbor discovery algorithm. Likewise, we can show that node r has not yet weakly terminated in $\beta(\mathcal{Q}_i)|t_i$.

Observe that, by our construction of $\mathcal{D}(\mathcal{Q}_i)$, for node r to be able to distinguish between executions α and $\beta(\mathcal{Q}_i)$, execution $\beta(\mathcal{Q}_i)$ should contain a time-slot, which is after time-slot t_i , such that node r is in receive mode on channel c during that time-slot. Let t_{i+1} be the *first* such time-slot. Clearly, $t_{i+1} > t_i$. It can be easily verified that

$$\alpha \xleftrightarrow[t_{i+1}-1]{t_i} \beta(\mathcal{Q}_i).$$

We now show how to construct \mathcal{Q}_{i+1} such that the two properties, namely (P1) and (P2), and the two invariants, namely (I1) and (I2), are satisfied.

Note that node r hears a clear message during time-slot t_{i+1} if and only if exactly one node in \mathcal{Q}_i is in transmit mode on channel c . Otherwise, node r does not hear anything during time-slot t_{i+1} , in which case it cannot distinguish between executions α and $\beta(\mathcal{Q}_i)$ after time-slot t_{i+1} as well. Therefore, in the second case, we set $\mathcal{Q}_{i+1} = \mathcal{Q}_i$.

Now, consider the first case in which exactly one node in \mathcal{Q}_i , say s , is in transmit mode on channel c . Clearly, the view of node r during time-slot t_{i+1} in execution α is different from that in execution $\beta(\mathcal{Q}_i)$. We refer to such a time-slot during which node r is in receive mode on channel c and exactly one node from \mathcal{Q}_i is in transmit mode on channel c as a *critical* time-slot. To maintain indistinguishability, we have to remove node s from \mathcal{Q}_i . However, removing node s may cause some earlier time-slots in the execution $\beta(\mathcal{Q}_i)$ to turn critical. The nodes transmitting during such critical time-slots have to be removed from \mathcal{Q}_i as well. This may again cause some other time-slots (before the time-slot t_{i+1}) to turn critical, and so on. Note that this process of removing nodes from \mathcal{Q}_i has to eventually terminate because \mathcal{Q}_i contains only finitely many nodes. Let \mathcal{R}_i denote the set of nodes that have to

be removed from \mathcal{Q}_i . It can be shown using induction on u that, for each time-slot $u \leq t_{i+1}$,

$$\left(\text{view}(r, \alpha, u) = \text{view}(r, \beta(\mathcal{Q}_i \setminus \mathcal{R}_i), u) \right) \wedge \left(\forall s : s \in \mathcal{Q}_i \setminus \mathcal{R}_i : \text{view}(s, \beta(\mathcal{Q}_i), u) = \text{view}(s, \beta(\mathcal{Q}_i \setminus \mathcal{R}_i), u) \right). \quad (1)$$

We set $\mathcal{Q}_{i+1} = \mathcal{Q}_i \setminus \mathcal{R}_i$. From (1), it follows that

$$\alpha \xleftrightarrow[t_{i+1}]{r} \beta(\mathcal{Q}_{i+1}).$$

Observe that the invariant (I1) is clearly satisfied by our choice of \mathcal{Q}_{i+1} and t_{i+1} . It remains to be shown that invariant (I2) also holds. To establish (I2), we actually prove a stronger invariant. Let $\psi(\mathcal{Q}_i, t_i)$ denote the number of time-slots in the sub-execution $\beta(\mathcal{Q}_i)|t_i$ during which node r is in receive mode on channel c and a non-empty subset of nodes of \mathcal{Q}_i are in transmit mode on channel c . Note that the subset size has to be at least two by our construction. Clearly, $\psi(\mathcal{Q}_0, t_0) = 0$. We prove that our construction satisfies the following invariant:

$$|\mathcal{Q}_i| \geq w - i + \psi(\mathcal{Q}_i, t_i). \quad (13)$$

The invariant (I2) follows from the invariant (13) because $\psi(\mathcal{Q}_i, t_i) \geq 0$ for each $i \leq w$. We next show that our construction satisfies the invariant (13).

Lemma 2. Consider the sequences $\mathcal{Q}_0, \mathcal{Q}_1, \dots, \mathcal{Q}_w$ and t_0, t_1, \dots, t_w constructed as described above. For each $i \leq w$, \mathcal{Q}_i and t_i satisfy the invariant (13).

Proof. The proof is by induction on i . For $i = 0$, we have

$$\begin{aligned} |\mathcal{Q}_0| &= w \\ &= w - 0 + 0 \\ &= w - 0 + \psi(\mathcal{Q}_0, t_0). \end{aligned}$$

Now, assume that the invariant holds for \mathcal{Q}_i and t_i for $i < w$. We have to show that the invariant holds for \mathcal{Q}_{i+1} and t_{i+1} . There are two cases to consider:

- Case 1 ($\mathcal{Q}_{i+1} = \mathcal{Q}_i$): In this case, $\psi(\mathcal{Q}_{i+1}, t_{i+1}) \leq \psi(\mathcal{Q}_i, t_i) + 1$. We have

$$\begin{aligned} |\mathcal{Q}_{i+1}| &= |\mathcal{Q}_i| \\ &\geq w - i + \psi(\mathcal{Q}_i, t_i) \\ &= w - (i + 1) + (\psi(\mathcal{Q}_i, t_i) + 1) \\ &\geq w - (i + 1) + \psi(\mathcal{Q}_{i+1}, t_{i+1}). \end{aligned}$$

- Case 2 ($\mathcal{Q}_{i+1} = \mathcal{Q}_i \setminus \mathcal{R}_i$, for some non-empty \mathcal{R}_i): Let $a = |\mathcal{R}_i|$. Let b denote the number of time-slots that turn critical (directly or indirectly) due to the removal of the first node from \mathcal{Q}_i . Clearly, $\psi(\mathcal{Q}_{i+1}, t_{i+1}) = \psi(\mathcal{Q}_i, t_i) - b$. Every node removed from \mathcal{Q}_i except the first node reduces the value of ψ by at least one. Therefore, $a - 1 \leq b$. We have

$$\begin{aligned} |\mathcal{Q}_{i+1}| &= |\mathcal{Q}_i| - a \\ &\geq w - i + \psi(\mathcal{Q}_i, t_i) - a \\ &= w - i + \psi(\mathcal{Q}_{i+1}, t_{i+1}) + b - a \\ &\geq w - i + \psi(\mathcal{Q}_{i+1}, t_{i+1}) - 1 \\ &= w - (i + 1) + \psi(\mathcal{Q}_{i+1}, t_{i+1}). \end{aligned}$$

Therefore the invariant holds in both cases. \square

It follows from our construction that:

Lemma 3. The execution α contains at least w time-slots during which node r is in receive mode on channel c before the algorithm NDA can terminate locally at node r .

A node has a single transceiver. This implies that the set of time-slots during which a node is in receive mode on channel c is disjoint with those during which it is in receive mode on channel d , where $c \neq d$. As a result, we have:

Theorem 4. Let r_o denote a node with the largest exclusive channel set among all nodes in the network, and let $x = |\mathcal{X}(r_o)|$. Then node r_o has to be in receive mode during at least $(n - p) \times x$ time-slots before any deterministic neighbor discovery algorithm can terminate locally at r_o .

Finally, we have the following theorem:

Theorem 5. Consider a trivial configuration consisting of a single node whose channel availability set is the universal channel set. Then any deterministic neighbor discovery algorithm has to use at least $(n - 1) \times m$ time-slots before the algorithm can terminate locally at the node.

Note that the lower-bound proof can be potentially simplified by only considering *star* configurations (nodes are arranged in the form of a star). However, the lower-bound proof presented here is stronger in the sense that it establishes the minimum number of time-slots that a deterministic neighbor discovery algorithm has to use for any configuration. Also, note that, as long as $n - p = \Theta(n)$, the time-complexity of any deterministic neighbor discovery algorithm cannot be sublinear in n .

4. Fast neighbor discovery algorithms when nodes are collision-aware

In this section, we describe two fast deterministic algorithms for neighbor discovery that satisfy the strong termination condition when nodes can detect collisions. Specifically, nodes can not only detect that the neighbor discovery has completed at all nodes, but they also terminate simultaneously.

4.1. The main idea

The core of our neighbor discovery algorithm is a *verification* algorithm that allows nodes to *test* that the network-size is upper-bounded by a certain value. Specifically, all nodes independently estimate an upper-bound for the network-size – using some deterministic rule – and attempt to verify that it is correct. The verification algorithm satisfies the following two properties: *agreement* and *integrity*. The agreement property ensures that all nodes agree on whether their current estimate for an upper-bound on the network-size is correct. The integrity property ensures that a node believes its estimate for an upper-bound on the network-size to be correct if and only if the number of nodes in the network is indeed upper-bounded by the estimated value. Observe that n is a trivial upper-bound on network size p . However, to obtain a fast algorithm, nodes need to know a *tighter* upper-bound on the network-size since p may be much smaller than n .

The execution of the neighbor discovery algorithm is divided into *phases* of geometrically increasing length. In each phase, nodes execute an instance of the verification algorithm. The next phase is executed only if the current instance of the verification algorithm indicates that the estimate of the upper-bound on the network-size is incorrect.

For ease of exposition, we first describe the verification algorithm assuming that the entire network operates on a single channel. We later describe how our algorithm can be extended when the network may operate on multiple channels and, moreover, different nodes may have different channels in their availability sets.

4.2. Verifying the network-size assuming a single channel

In our verification algorithm, we make extensive use of *information by silence* [5]. If a node does not receive any signal for a certain number of time-slots, then it can use this fact to make certain deduction. Suppose nodes wish to verify that there are at most x nodes in the network. The verification algorithm consists of three stages. In the first stage, we run a leader election algorithm. Since all nodes participate in the verification algorithm and no node knows the actual network diameter, we do not guarantee a unique leader in the network. However, we ensure that any two leaders are sufficiently apart so that their activities in the remaining two stages do not interfere with each other. We do ensure that there is a unique leader in the network if there are at most x nodes in the network. In the second stage, a leader explores its neighborhood using a token-based traversal scheme. The token keeps track of the number of nodes that it has visited so far during the neighborhood traversal. At any time, if a node currently holding the token detects that there are more than x nodes in the network, then it immediately aborts the traversal. Otherwise, the traversal completes successfully and the token returns to the initiating leader. Finally, in the third stage, if the neighborhood traversal in the second stage completed successfully, then the leader informs all nodes in the network about the successful completion. On the other hand, if the traversal was unsuccessful, that is, it was aborted by some node, then no transmission occurs in the third stage. This ensures that a non-leader node receives a signal – either a clear message or collision noise – if and only if there are at most x nodes in the network. We now describe each stage in more detail. We refer to the set of nodes that are at a distance of at most h hops from a node as its h -hop neighborhood.

4.2.1. First stage: Leader election

Chrobak et al. [5] describe a deterministic algorithm for electing a leader in a radio network when the label space size is within a linear factor of the network-size. However, in our system model, the label space size may be much larger than the network-size and no node knows the network-size. As a result, it is hard, if not impossible, to elect a unique leader in the network in $O(\text{poly}(p)\text{polylog}(n))$ time. We use a leader election algorithm that does not guarantee a unique leader in the network but, nevertheless, ensures that any two leaders are separated by at least $2x + 2$ hops. This, in turn, guarantees that neighborhood traversals undertaken by two leaders concurrently in the second stage do not interfere with each other.

Our leader election algorithm consists of $\log n$ rounds and each round consists of $2x + 2$ time-slots. Each node is in either an *active* or *passive* state. All nodes are active initially. Once a node becomes passive, it stays passive. Intuitively, a node is a candidate for a leader as long as it is active and is no longer a candidate once it becomes passive. We ensure that, after round i , there are at most $\frac{n}{2^i}$ active nodes in any connected sub-network whose diameter is at most $2x + 2$ hops. Clearly, after $\log n$ rounds, any connected sub-network with diameter at most $2x + 2$ hops contains at most one active node which then elects itself as a leader.

We maintain the property that if two nodes within a distance of $2x + 2$ hops from each other are still active at the end of round i then they have the same i most significant bits in their labels. Clearly, the property holds at the beginning. Assume that the property holds after $i - 1$ rounds with $1 \leq i < \log n$. Consider a node r that is still active at the beginning of round i . Let $\text{msb}(r, i)$ denote the i th most significant bit in the label of node r . There are two cases depending on whether $\text{msb}(r, i)$ is 1 or 0. If $\text{msb}(r, i)$ is 1, then r stays active at the end of round i . Further, it informs all nodes in its $(2x + 2)$ -hop neighborhood about its existence using simple *flooding*. On the other hand, if $\text{msb}(r, i)$ is 0, then r stays active at the end of round i if and only if there is no active node s in its $(2x + 2)$ -hop neighborhood with $\text{msb}(s, i)$ as 1.

Flooding. The flooding works as follows. Let r be a node that is active at the beginning of round i such that $\text{msb}(r, i)$ is 1. Node r broadcasts a message in the first time-slot of round i . If a node s receives some signal – a clear message or collision noise – in time-slot t , where $1 \leq t \leq 2x + 1$, then s broadcasts a message in time-slot $t + 1$ provided s has not transmitted earlier during round i . It can be verified that every node in $(2x + 2)$ -hop neighborhood of node r , except possibly r , receives a signal during some time-slot of round i .

A formal description of the first stage is given in Fig. 2. Let $\text{neighbor-set}(r, h)$ denote the set of nodes that are within a distance of h hops from node r . Let $\text{prefix}(r, i)$ denote the prefix of length i in the label of node r . We define $\text{prefix}(r, 0)$ to be the empty prefix. Finally, let $\text{active-set}(r, h, i)$ denote the subset of nodes in $\text{neighbor-set}(r, h)$ that are still active at the end of round i . Since all nodes are active initially, we define $\text{active-set}(r, h, 0) = \text{neighbor-set}(r, h)$. We have:

Lemma 6. *Let r_0 be a node with the largest label among all nodes in $\text{neighbor-set}(r_0, 2x + 2)$. For each i with $0 \leq i \leq \log n$, we have*

$$r_0 \in \text{active-set}(r_0, 2x + 2, i)$$

and

$$r \in \text{active-set}(r_0, 2x + 2, i) \Rightarrow \text{prefix}(r, i) = \text{prefix}(r_0, i).$$

Proof. The proof is by induction on i . \square

It follows that:

Lemma 7. *After the end of the first stage, the distance between any two leaders is at least $2x + 2$ hops.*

Proof. Consider a node r and let r_0 be the node with the largest label among all nodes in $\text{neighbor-set}(r, 2x + 2)$. If node r is elected as a leader, then $r \in \text{active-set}(r_0, 2x + 2, \log n)$. From Lemma 6, this implies that $\text{prefix}(r, \log n) = \text{prefix}(r_0, \log n)$. In other words, nodes r and r_0 have the same label, implying that $r = r_0$.

Now, assume that nodes r and s are elected as leaders after $\log n$ rounds and, on the contrary, distance between r and s is less than $2x + 2$ hops. As argued before, nodes r and s have the largest labels among all nodes in their respective $(2x + 2)$ -hop neighborhoods. Since $r \in \text{neighbor-set}(s, 2x + 2)$, this implies that nodes r and s have the same labels. \square

The diameter of a network is bounded by its size. Therefore, it follows from the previous lemma that:

Corollary 8. *If the network contains at most x nodes, then there is exactly one leader in the network.*

4.2.2. Second stage: Neighborhood traversal

In this stage, each node elected as a leader in the first stage explores its neighborhood in a depth first manner using a token-based traversal scheme. Kowalski and Pelc [14] describe a token-based depth first traversal algorithm for a radio network when nodes are not collision-aware and the label space size is within a linear factor of the network-size. However, in our system model, nodes are collision-aware and the label space size may be much larger than the network-size. The main idea behind depth first traversal is as follows [6,14]. The leader holds the token initially. A node currently holding the token tries to locate a neighbor that has not been visited so far. If no such neighbor exists, then it returns the token to the node from which it received the token in the first place. Otherwise, it transfers the token to one of its neighbors that has not been visited by the token yet. A token contains an integer that keeps track of the number of nodes it has visited so far and the traversal is aborted as soon as the value of the integer exceeds

```

First stage at node  $r$ :

// initially all nodes are active
status := active;
for  $i := 1$  to  $\log n$  do
    // execute round  $i$ 
    done := false;
    if (status = active) and (ith most significant bit is 1) then transmit := true;
    else transmit := false; endif;

    for  $j := 1$  to  $2x + 2$  do
        if not(done) then
            if (transmit) then
                transmit an empty message;
                done := true;
            else
                listen;
                if (heard a signal) then transmit := true; endif;
            endif;
        else stay quiet endif;
    endfor;

    if (status = active) and (ith most significant bit is 0) and (transmit) then
        // there is an active node with larger identifier in the neighborhood
        status := passive;
    endif;
endif;

if (status = active) then leader := true;
else leader := false; endif;

```

Fig. 2. Leader election in the first stage.

x . It is convenient to view depth first traversal as consisting of multiple *steps*, where, in each step, the token either moves *forward* to an unvisited node or *backward* to an already visited node. Each step consists of at most $2 \log n + 4$ time-slots. Although different steps may contain different number of time-slots, it is convenient to assume that all steps contain exactly $2 \log n + 4$ time-slots.

We now describe a step. If a node has already been visited by the token, then we say that the node is *tagged*; otherwise, we say that the node is *untagged*. Suppose the token is currently being held by node r . In the first time-slot of the step, r broadcasts a message to determine if it has any untagged neighbor. A neighbor of node r , on receiving such a message, transmits a message in the second time-slot of the step if it is untagged. There are three cases to consider depending on what node r hears during the second time-slot of the step:

- *Case 1* (r does not hear anything): In this case, node r has no untagged neighbor. Therefore, in the third time-slot of the step, node r returns the token back to the node from which it received the token.
- *Case 2* (r hears a clear message): In this case, node r has exactly one untagged neighbor. In the third time-slot of the step, node r sends the token to that untagged neighbor.
- *Case 3* (r hears collision noise): In this case, node r has two or more untagged neighbors. In the third time-slot of the step, it informs all its untagged neighbors that there are many of them and contention between them has to be resolved. We describe a contention resolution scheme that node r can use to select an untagged neighbor with the largest label. Once the contention is resolved, node r sends the token to the selected neighbor in the next time-slot.

Contention resolution. We resolve contention between untagged neighbors of node r using $2 \log n$ time-slots. There are two types of time-slot: *normal* and *feedback*. Contention resolution consists of alternating sequence of normal and feedback time-slots. In a normal time-slot, a certain subset of untagged neighbors transmit whereas node r listens. In a feedback time-slot, the reverse happens; that is, node r transmits and its untagged neighbors listen. Intuitively, the contention resolution between untagged neighbors of node r is similar to that between nodes during leader election. For convenience, we refer to a pair of normal time-slot followed by a feedback time-slot as a round.

Each untagged neighbor is in either *competing* or *quiescent* state. Initially, all untagged neighbors are in competing states. Once a neighbor becomes quiescent, it stays quiescent. Intuitively, an (untagged) neighbor is in contention as long as it is in competing state; the node is eliminated from contention once it becomes quiescent. The rules governing transition from competing to quiescent state are similar to those governing transition from active to passive state used in leader election algorithm. One of the differences is that the round now consists of only two time-slots instead of $2x + 2$ time-slots. In the first (normal) time-slot of round i , all those untagged neighbors of node r that are still in competing state at the beginning of round i transmit if their i th most significant bit is 1. Node r transmits in the second (feedback) time-slot of round i if it hears some signal during the first time-slot. At the end of round i , all those untagged neighbors of node r that were still in competing state at the beginning of round i become quiescent if they hear a transmission during the second time-slot and their i th most significant bit is 0.

A formal description of the second stage is given in Figs. 3 and 4. Observe that, since a traversal is aborted as soon as more than x

```

Second stage at node  $r$ :

 $parent$  : node from which the token is received;

if ( $leader$ ) then
    initialize the token with count of 1;
     $tagged := true$ ;
else  $tagged := false$ ; endif

for  $i := 1$  to  $2x$  do
    // first time-slot of step  $i$ 
    if (have the token) then
        transmit "searching for an unvisited neighbor";
    else listen and update the set of neighbors; endif;

    // second time-slot of step  $i$ 
    if not( $tagged$ ) and (heard a transmission during the first time-slot) then
        transmit "waiting for the token";
    else listen; endif;

    // third time-slot of step  $i$ 
     $done := true$ ;
    if (have the token) then
        if (heard silence during the second time-slot) then
            if not( $leader$ ) then
                transmit "returning the token to  $parent$ ";
            endif;
        else if (heard clear transmission during the second time-slot) then
             $s :=$  the node from the clear transmission was received;
            transmit "transferring the token to  $s$ ";
        else
            transmit "contention resolution required";
             $done := false$ ;
        endif;
    else
        listen and receive the token, if applicable;
        if not( $tagged$ ) and (token holder requests contention resolution) then
             $done := false$ ;
        endif;
    endif;

    // remaining  $2 \log n + 1$  time-slots of step  $i$ 
    if not( $done$ ) then
        participate in contention resolution for  $2 \log n$  time-slots; // logically similar to the first stage
        if (have the token) then
             $s :=$  node that won contention resolution;
            transmit "transferring the token to  $s$ ";
        else if (won contention resolution) then
            listen and receive the token;
        endif;
        else stay quiet for the next  $2 \log n + 1$  time-slots; endif;
    endif;
endfor;

```

Fig. 3. Depth first traversal in the second stage.

nodes are detected, the traversal involves only nodes in the x -hop neighborhood of a leader. We define the distance between two sets of nodes as the *smallest* distance between nodes from the two sets.

Lemma 9. *Neighborhood traversals initiated by two different leaders involve mutually exclusive sets of nodes. Further, the two sets of nodes are separated by at least two hops.*

Proof. A token during depth first traversal travels a distance of at most x hops from its initiating leader. Further, from Lemma 7, two leaders are separated by at least $2x + 2$ hops. \square

Observe that the x -hop neighborhood of a leader contains more than x nodes if and only if the network contains more than x nodes. Therefore, we have:

Lemma 10. *The neighborhood traversal by a leader completes successfully if and only if the network contains at most x nodes.*

Proof. Neighborhood traversal initiated by a leader is aborted if and only if its x -hop neighborhood contains more than x nodes. \square

As far as time-complexity of neighbor traversal is concerned, there are at most x forward steps and at most x backward steps.

```

Second stage at node  $r$  (continued):

On receiving the token:
  increment the count in the token by 1;
  if (count in the token has exceeded  $x$ ) then
    destroy the token;
  else
    tagged := true;
    parent := node from which the token is received;
  endif;

After  $4x(\log n + 2)$  time-slots have elapsed:
  if (leader) and (have the token) then
    result := success;
  else result := failure; endif;

```

Fig. 4. Depth first traversal in the second stage (continued).

Therefore the second stage consists of at most $2x \cdot (3 + 2 \log n + 1) = 4x(\log n + 2)$ time-slots.

4.2.3. Third stage: Result dissemination

If the neighborhood traversal initiated by a leader in the second stage completes successfully (that is, the token returns to its initiating leader), then it follows from Lemma 10 that there at most x nodes in the network. Further, it follows from Corollary 8 that there is exactly one leader in the network. Therefore this (unique) leader can safely infer that its current estimate for an upper-bound on the network-size is indeed correct. However, this knowledge still has to be disseminated to the rest of the nodes in the network. This can be accomplished in $x - 1$ time-slots using simple flooding similar to the one described in the first stage. Only $x - 1$ time-slots are required because the network contains at most x nodes and, as a result, every node can be reached from the leader using at most $x - 1$ hops. On the other hand, if the neighborhood traversal initiated by a leader in the second stage is aborted, then the leader can safely infer that its current estimate for an upper-bound on the network-size is incorrect and does not transmit any message in the third stage. It can be easily verified that a non-leader node receives a signal – either a clear message or collision noise – during some time-slot in the third stage if and only if its current estimate for an upper-bound on the network-size is correct. Combining this with Lemma 10, we obtain the following two lemmas; a formal description of the third stage is given in Fig. 5.

Lemma 11. Assume that the network contains at most x nodes. Then, at the end of third stage, all nodes believe their current estimate for an upper-bound on the network-size to be correct.

Lemma 12. Assume that the network contains more than x nodes. Then, at the end of the third stage, all nodes believe their current estimate for an upper-bound on the network-size to be incorrect.

4.2.4. Complexity of verification

We analyze the time-complexity of the verification algorithm. The time-complexity of the first stage is $(2x + 2) \log n$ time-slots, the second stage is $4x(\log n + 2)$ time-slots and the third stage is $x - 1$ time-slots. Therefore, the overall time-complexity of the verification algorithm is $6x \log n + 9x + 2 \log n - 1 = O(x \log n)$ time-slots.

```

Third stage at node  $r$ :

if (leader) and (result = success) then
  // propagate success to all nodes in the network
  transmit := true;
else transmit := false; endif;

done := false;
for  $i := 1$  to  $x - 1$  do
  if not(done) then
    if (transmit) then
      transmit an empty message;
      done := true;
    else
      listen;
      if (heard a signal) then transmit := true; endif;
    endif;
  else stay quiet; endif;
endfor;

if (transmit) then
  // estimate on upper bound is correct;
  terminate the algorithm;
else
  // estimate on upper bound is incorrect;
  execute the next phase;
endif;

```

Fig. 5. Result dissemination in the third stage.

4.3. Neighbor discovery

As discussed earlier, the neighbor discovery algorithm consists of multiple phases. In phase k with $k \geq 0$, all nodes execute an instance of the verification algorithm with an estimate value of 2^k . The algorithm terminates once the current instance of the verification algorithm indicates that the estimate is correct. Clearly, there are $\lceil \log p \rceil + 1$ phases. Neighbor discovery itself can be done during the second stage of the verification algorithm when neighborhood traversal is performed. Clearly, in the last phase, exactly one node is elected as a leader, implying that there is exactly one token in the network. Recall that, in the first time-slot of a step, only the node holding the token is in transmit mode and all other nodes are in receive mode. During this time-slot, all neighbors of the node holding the token will receive its transmission successfully. Since, in the last phase, the token visits each node at least once, each node eventually discovers all its neighbors. The overall time-complexity of neighbor discovery is therefore given by

$$\begin{aligned}
 & \sum_{k=0}^{\lceil \log p \rceil} (6 \cdot 2^k \cdot \log n + 9 \cdot 2^k + 2 \cdot \log n - 1) \\
 &= 6 \cdot (2^{\lceil \log p \rceil + 1} - 1) \cdot \log n + 9 \cdot (2^{\lceil \log p \rceil + 1} - 1) \\
 & \quad + 2 \cdot (\lceil \log p \rceil + 1) \cdot \log n - (\lceil \log p \rceil + 1) \\
 &= (12 \log n + 18) \cdot 2^{\lceil \log p \rceil} + (2 \log n - 1) \lceil \log p \rceil \\
 & \quad - (4 \log n + 10) \\
 &= O(p \log n). \tag{2}
 \end{aligned}$$

We denote our neighbor discovery algorithm described in this section by NDA_SC.

Theorem 13. NDA_SC satisfies no false discovery, eventual discovery and simultaneous termination properties. Further, its time-complexity is $O(p \log n)$.

Fig. 6. Variation in the time-complexity with the (a) network-size, and (b) label size.

4.4. Handling multiple channels

We use NDA_SC, which assumes that all nodes in the network have a single channel in their channel availability sets, to solve the neighbor discovery problem when different nodes in the network may have different channel availability sets. The main idea is to *simulate* each action of NDA_SC as if the entire network is operating over some single *virtual* channel c_0 such that:

Nodes r and s are neighbors on virtual channel c_0 in the simulated network if and only if nodes r and s are neighbors on some channel in \mathcal{UCS} in the real network.

The simulation involves replacing each time-slot t in NDA_SC with a *frame* consisting of m time-slots, denoted by $frame(t)$; the i th time-slot in the frame corresponds to the channel c_i . Any transmit (respectively, listen) action by a node r during a time-slot t in NDA_SC is mapped to multiple transmit (respectively, listen) actions by r as follows: r transmits (respectively, listens) during the i th time-slot of $frame(t)$ if the channel c_i is in its availability set. The view of node r at the end of $frame(t)$, if r is in receive mode, for the purpose of simulating NDA_SC is given by

- *silence*: if node r neither receives a clear message nor hears collision noise during any time-slot of $frame(t)$,
- *clear transmission*: if node r receives a clear message from exactly one node during all time-slots of $frame(t)$ and, moreover, does not hear collision noise during any time-slot of $frame(t)$,
- *collision*: if node r either hears collision noise during at least one time-slot of $frame(t)$ or receives a clear message from at least two different nodes during two different time-slots of $frame(t)$.

We refer to the algorithm obtained after replacing each time-slot by a frame as explained above by NDA_MC. Intuitively, if all nodes in the network were indeed operating on the single channel c_0 and NDA_SC was being used for neighbor discovery, then the state of a node at the end of time-slot t in NDA_SC will be “same” as the state of the node at the end of $frame(t)$ in NDA_MC. From the discussion in Section 4.3, for every node in the network, there exists at least one time-slot during the execution of NDA_SC in which the node transmits and all other nodes listen. This implies that, for every node in the network, there exists at least one *frame* during the execution of NDA_MC in which the node transmits and other nodes listen. It is during this frame that a node is guaranteed to be discovered by each of its neighbors on their mutually common channels. Clearly, the time-complexity of NDA_MC is m times that of NDA_SC, and is given by

$$\left\{ (12 \log n + 18) \cdot 2^{\lceil \log p \rceil} + (2 \log n - 1) \cdot \lceil \log p \rceil - (4 \log n + 10) \right\} \cdot m. \quad (3)$$

Therefore, we have

Theorem 14. *NDA_MC satisfies no false discovery, eventual discovery and simultaneous termination properties. Further, its time-complexity is $O(p m \log n)$.*

4.5. Experimental results

We simulated the neighbor discovery algorithm NDA_MC in Java to evaluate its performance experimentally. Note that we do not need to use simulation tools such as NS-2 since our algorithms are deterministic and there is no randomness. Thus, for a given scenario consisting of node placements, node label assignments and node channel assignments, there is only one outcome, and the behavior of all nodes will be the same even if we run the experiment many times. For the evaluation, we used the following parameters:

- Deployment area: 400 m × 400 m
- Size of universal channel set m : 20
- Size of label space n : 2^{16}
- Communication range: 100 m.

When generating networks for a given network-size, we only considered those networks that were connected. We measured the time-complexity of the neighbor discovery algorithm in terms of number of frames as the number of nodes in the network increased from 1 to 400. The results are shown in Fig. 6(a).

The time-complexity measured experimentally exactly matches the time-complexity computed analytically in (3). The graph resembles a step function because the time-complexity actually depends on $\lceil \log p \rceil$ rather than $\log p$. For example, the time-complexity for $p = 10$ is exactly same as the time-complexity for $p = 16$ because $\lceil \log 10 \rceil = \lceil 3.32 \rceil = 4 = \lceil \log 16 \rceil$. The time-complexity of the neighbor discovery algorithm only depends on three factors (p , n and m), and the time-complexity does not depend on factors such as dimensions of deployment area, network density, network degree and communication range of a node. Our experiments verified this.

We then measured the relationship between the label size and the number of frames needed, and the results are shown in Fig. 6(b). The relationship between the label size and the number frames has been measured for three different values of p . When the value of p goes up, the frame size goes up as well, and this is shown Fig. 6(b).

The simulation experiments were also used to experimentally verify the correctness of the algorithm. For each execution, the correctness was verified by manually checking and by a simple centralized software that has global knowledge. This step experimentally shows that the algorithm is error free and the implementation is bug free.

