More the Merrier: Neighbor Discovery on Duty-cycled Mobile Devices in Group Settings

Reynaldo Morillo, *Student Member, IEEE,* Yanyuan Qin, *Student Member, IEEE,* Alexander Russell, Bing Wang, *Senior Member, IEEE*

***Abstract*—Neighbor discovery on duty-cycled mobile devices in group settings arises in many applications. In such scenarios, it is sufficient for an arbitrary node in a group to discover a new node. While pairwise neighbor discovery schemes can be directly applied to group settings, their performance can be severely limited as they are not designed to coordinate the efforts of group members. Explicit coordination among the group members, however, can incur large overhead in mobile networks, where the group membership changes dynamically over time. In this paper, we focus on schemes that require no explicit communication among the group members, and nodes follow deterministic schedules that can be succinctly represented. We first define the notion of *ideal duty cycle* for a group, and then develop two deterministic neighbor discovery schemes for group settings, and show that both of them achieve effective duty cycle close to the ideal duty cycle. In addition, we show that the schemes are lightweight and easy to implement using experiments in a testbed. Last, we use a case study to demonstrate the usage of our proposed schemes and show that a simple enhancement leveraging the deterministic nature of the schemes leads to significant performance improvement, at the cost of only slight extra overhead.**

***Index Terms*—Wireless networks, mobile networks, network management, neighbor discovery**

I. Introduction

Neighbor discovery, i.e., finding neighboring nodes that are in the communication range of each other, is a fundamental step for many functions in wireless networks such as route determination, data relaying, or simply maintaining local network structure. In mobile wireless networks, neighbor discovery needs to be carried out on a continuous basis, since a node's neighbors change over time due to node mobility. We focus on *asynchronous* discovery for *duty-cycled* mobile devices, i.e., the wireless nodes do not have synchronized clocks and have to discover each other with no central coordination (e.g., through a shared server or dedicated control channel), and they are duty-cycled to conserve limited battery resources.

Pairwise asynchronous neighbor discovery between two duty-cycled nodes has been studied extensively (see Section II). In practice, however, neighbor discovery often arises in group settings. For example, suppose a message needs to be broadcasted in a mobile delay/disruption tolerant network (DTN) [14]. A group of nodes, $\mathcal{A}$, have already received a copy of the message, while a node B has not yet received the message. Then it is sufficient that a node $A \in \mathcal{A}$ discovers B, since after that, A can forward the message to B. The key requirement of neighbor discovery in group settings is that an arbitrary node in the group $\mathcal{A}$ discovers B. Our goal is to minimize the *discovery latency*, i.e., the delay for the first node in $\mathcal{A}$ to discover B.

Pairwise neighbor discovery schemes can be applied directly to group settings. Specifically, each node $A \in \mathcal{A}$ can use a pairwise scheme to discover B; then the discovery latency is simply the minimum delay of all the pairwise discovery cases, i.e., between each node in $\mathcal{A}$ and B. Pairwise strategies, however, can lead to poor performance in group settings as they may not effectively coordinate the schedules of distinct group nodes.

In this paper, we investigate asynchronous neighbor discovery in group settings, taking advantage of the multiple nodes in a group. We consider both *symmetric* (or *homogeneous*) neighbor discovery, where nodes have the same duty cycle, and *asymmetric* (or *heterogeneous*) neighbor discovery where nodes can have different duty cycles. Our focus is on developing *deterministic* algorithms, where the schedules of the nodes are determined beforehand.

Such schedules have two major advantages: (i) the worst-case discovery latency is deterministic, and (ii) once two nodes know the parameters of each other's schedules, they can precisely predict each other's future waking times, allowing convenient communications in the future, which can lead to significantly improved application performance (see Section VII). Our study makes the following three main contributions:

- We define the notion of *ideal duty cycle* for a group, which is used to gauge the *effective duty cycle* of a group (i.e., the fraction of time when at least one node in the group is awake). We show that using pairwise schemes in a naive manner can lead to an effective duty cycle far below the ideal duty cycle. We further show that a simple randomized algorithm achieves the ideal duty cycle, which will be used as a baseline to compare with the deterministic algorithms that we develop.
- We develop two deterministic algorithms, both achieving effective duty cycle close to the ideal duty cycle. Specifically, we propose a random shift technique that can be applied to existing pairwise schedules, while maintaining the desirable properties of such schedules and ensuring that the effective duty cycle of the group approximates the ideal duty cycle with high probability.

Manuscript received xx xx, 2021.

R. Morillo, Y. Qin, A. Russell and B. Wang are with the Computer Science & Engineering Department at the University of Connecticut. This research was supported in part by the National Science Foundation under grants 1717432 and 1925706. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

We further propose a new family of "modular polynomial" based deterministic schedules, which provide a rigorous guarantee on expected discovery time (scaling in ideal duty cycle) in group settings, and can still be succinctly described. Both schemes are lightweight and easy to implement (as shown in our testbed experiments), requiring no explicit communication among the nodes in the group to coordinate the discovery of a new node.

- As a case study, we use our proposed schemes as basic building blocks in a practical application. Our results show that a simple enhancement leveraging the deterministic nature of our schedules leads to substantial improvement in application-level performance, significantly outperforming the baseline randomized algorithm.

The rest of the paper is organized as follows. Section II briefly describes related work. Section III describes the background and the problem setting. Sections IV and V present the random shift and polynomial techniques, respectively. Section VI presents the testbed setting and compares the experimental results from the testbed with those from the simulations. Section VII uses a case study to illustrate the application of our proposed schemes. Last, Section VIII concludes the paper and presents future work.

II. Related Work

We briefly review existing work on neighbor discovery in the following four categories: (i) neighbor discovery in group settings when nodes are not duty-cycled, (ii) neighbor discovery in pairwise settings when nodes are duty cycled, (iii) neighbor discovery in group settings when nodes are duty cycled (i.e., the setting in our study), and (iv) other related work.

Neighbor discovery in group settings (not duty-cycled). The studies in [2], [3], [22], [28], [32] consider neighbor discovery in group settings. They, however, do not consider duty-cycled nodes, and hence the schedules mainly determine when a node needs to transmit (the node listens for the rest of the time). When a node is duty-cycled, a schedule needs to determine when the node is awake (it can only transmit/listen when it is awake) to conform to the duty cycle, while still ensure short discovery latency, which is the focus of this paper.

Neighbor discovery in pairwise settings (duty-cycled). Most existing studies on neighbor discovery with duty-cycled nodes consider pairwise settings (e.g., [6], [10], [12], [22], [40]). A recent study [30] broadly classifies existing pairwise schemes into four categories: the first category includes randomized schemes; the other three categories are deterministic schemes, based on over-half occupation, rotation-resistent intersection, and coprime cycles, respectively. An example randomized scheme is the Birthday Protocol [22], which considers a period of n slots, randomly choose m slots as awake slots, and the rest of the $n-m$ slots as asleep slots. A scheme based on over-half occupation ensures that for a period of n slots, a node is awake in at least half of the slots. Searchlight [1] in essence leverages the above principle, while is carefully designed to reduce the number of awake slots (and hence the duty cycle). In schemes based on rotation-resistent intersection, two nodes arrange their awake slots so that these slots intersect despite the rotations of the schedules. The various schemes [12], [18], [23], [24], [31], [40] based on quorum and difference sets are in this category. Last, schedules [4], [10], [17] based on coprimes are constructed using coprimes as parameters. As an example, in Disco [10], for duty cycle d, a node selects two different primes p_1, p_2 so that $1/p_1 + 1/p_2 \approx d$ and wakes up in the slots that are multiples of p_1 or p_2.

In general, existing pairwise deterministic schemes provide discovery latency within $c/(d_1 d_2)$ slots for two nodes with duty cycles d_1 and d_2, respectively, where c is a constant that depends on a particular scheme. Our study focuses on neighbor discovery in group settings, instead of pairwise settings. The random shift technique that we develop (see Section IV) extends existing pairwise schemes to group settings.

Neighbor discovery in group settings (duty-cycled). The literature on neighbor discovery in group settings with duty-cycled nodes is very sparse. The study in [5] designs a reference mechanism: when a node A in a group $\mathcal{A}$ discovers another node B, A informs B the schedules of all the other nodes in $\mathcal{A}$; then B verifies, by waking up proactively, which nodes in $\mathcal{A}$ are its neighbors. The authors further introduce a selective reference mechanism so that A only informs B the schedules of a subset of nodes in $\mathcal{A}$ that are more likely to be B's neighbors. A recent study [34] proposes a selective reference mechanism that estimates physical proximity of nodes using XGBoost. Our study focuses on minimizing the delay for an arbitrary node in $\mathcal{A}$ to discover B, which is a pre-requisite for neighbor verification in [5], [34]. In our case study (Section VII), we explore a proactive transmission mechanism, where immediately after a node receives a copy of a message, it wakes up proactively to transmit a copy of the message to potentially nearby nodes that might not have received a copy to improve application performance.

The study in [8] considers a static sensor network, where a node starts in Init state until it finds most of its neighbors, and then moves to Normal state, where it performs continuous neighbor discovery. The authors propose schemes to decide a node's schedule during continuous neighbor discovery, which are based on the notion of *segment*, i.e., a set of connected nodes, where two nodes are connected if they are directly connected (within transmission range of each other) or have a path of directly connected nodes between them. The discovery of a new node in a segment is a joint task by all segment nodes, and the schedule of a segment node (and its duty cycle) is based on the degree of its in-segment neighbors. These schemes target static networks. They are not suitable for mobile networks, where segments change dynamically over time due to node mobility, and hence segments need to be determined continuously that can incur significant overhead and the dynamics in segments will lead to changes in nodes duty cycles. We focus on mobile networks and propose schemes that do not require determining group members or segments. In our schemes, nodes have fixed duty cycles and their coordination is implicit, through the initial randomness in the schemes.

The study in [26] proposes WiFlock, an energy efficient protocol that combines neighbor discovery and group maintenance using a collaborative beaconing mechanism for flocking (i.e., scattered mobile nodes occasionally come together for a period of time, e.g., during shipping). It requires group-wide clock synchronization and is specifically designed for flocking applications. Our proposed schemes are applicable to general mobile networks, and requires no group maintenance or clock synchronization.

Other studies. The schemes in [36], [37] reduce the energy consumption of existing neighbor discovery schemes. The study in [29] proposes a generic framework that incorporates existing deterministic protocols, and allows flexibility in adjusting parameters. In [27], [33], beacons are allowed to be sent in non-wakeup slots, which differ from the assumptions in this paper. The schemes in [19], [38] are for mobile duty-cycled devices, which however require clock synchronization. The study in [35] designs robust neighbor discovery techniques in the presence of strong interference. Location prediction is used in [20] for neighbor discovery in vehicular ad hoc networks. A recent study [11] proposes using cross-technology for neighbor discovery.

III. Motivating Applications and Problem Setting

In this section, we first present several motivating examples to illustrate the broad applications of neighbor discovery in group settings. We then formulate the problem, present a simple randomized algorithm (to be used as comparison baseline) and a high-level overview of the two deterministic algorithms that we propose.

A. Motivating Applications

Data transfer in DTNs. Consider a DTN, where nodes are mobile and the network does not have consistent connectivity. For example, a group of underwater sensors move in a lake to monitor the underwater wildlife. When one sensor observes an interesting event, it will send a message to a particular destination, e.g., a special mobile data collector. The message can be transmitted to other nodes and eventually to the destination. Suppose multiple copies of the message are allowed inside the network (each carried by one node in the network) to reduce the delivery latency. Any node carrying a copy of the message can transmit it to the destination, at which time the message forwarding process stops. Another application scenario is broadcast data transfer, e.g., a node needs to broadcast a message (e.g., an emergency alert) to the rest of the nodes in the network. This scenario is similar to the unicast scenario that is described earlier; the only difference is that the message forwarding stops when every node (instead of the destination node) gets a copy of the message.

Mobile data collection. Consider an animal tracking application [16], [39], where a group of animals each carries a wireless tag. When some animals come close to each other (e.g., to a water hole), they will exchange the messages with each other so that the tag on one animal stores the information from the tags carried by other animals. Later on, when one of the tags is close to a data collection station, it can "deposit" all the data that it stores to the station. Another similar application is emergency rescue in outdoor hiking [13], [15]. Suppose each hiker carries a wireless device. Two hikers sync each other's sequence of locations when meeting each other. As a result, one hiker's wireless device can carry the location data of many hikers. When any wireless device is close to a base station, it uploads all the stored location data to the base station. When a missing hiker needs to located, a rescue team can use the data stored in base stations to find the last location of the hiker to facilitate the rescue efforts.

B. Problem Formulation

Summarizing the applications described above, at time t, let $\mathcal{A}_t$ denote the set of nodes that have some data or a copy of a message to be transmitted to other nodes in the network. Neighbor discovery at any time t is only concerned about the nodes that are not yet in $\mathcal{A}_t$, which are to be discovered by nodes in $\mathcal{A}_t$, so that they can get a copy of the data or message from nodes in $\mathcal{A}_t$. Consider an arbitrary node $B \notin \mathcal{A}_t$, i.e., B does not yet have a copy of the data or the message. Let $\mathcal{A}_t(B) \subseteq \mathcal{A}_t$ be the group of nodes that are in the transmission range of B. Then it is sufficient that an arbitrary node $A \in \mathcal{A}_t(B)$ discovers B, and transfers a copy of the data or the message to B. Fig. 1(a) illustrates this scenario, where the green nodes are in $\mathcal{A}_t$, while the blue nodes are not in $\mathcal{A}_t$. The circle surrounding node B represents its transmission range. For ease of exposition, we assume symmetric transmission range, i.e., if A is in the transmission range of B, then B is also in the transmission range of A. Then $\mathcal{A}_t(B) = \{A_1, A_2, A_3\}$ in the example in Fig. 1(a) and it is sufficient for one of the nodes in $\mathcal{A}_t(B)$ to discover B. Note that the group of nodes in $\mathcal{A}_t(B)$ do not have to be in the communication range of each other, or be aware that they are in B's group. Since the nodes are mobile, the nodes in B's group will change dynamically over time. Suppose that B is discovered at time t', then it is added to $\mathcal{A}_{t'}$. In Fig. 1(a), we further illustrate another instance of group-based discovery for blue node $B' \notin \mathcal{A}_t$, where it is sufficient for one green node in $\mathcal{A}_t(B') = \{A_3, A_4, A_5, A_6\}$ to discover B'. Note that A_3 is in both $\mathcal{A}_t(B)$ and $\mathcal{A}_t(B')$ at time t. When A_3 discovers B, it does not necessarily discover B', and vice versa.

In general, we formulate the problem of neighbor discovery in group settings as follows. Consider a group of nodes $\mathcal{A} = \{A_1, \ldots, A_k\}$, node $B \notin \mathcal{A}$, and all nodes in $\mathcal{A}$ are in the transmission range of B. The critical quantity is the time for the first node in $A \in \mathcal{A}$ to discover B, which represents the *discovery latency* in group settings. The goal is to choose the waking-and-sleeping schedules for all the nodes (including both nodes in $\mathcal{A}$ and node B) so that the discovery latency is minimized. For ease of notation, we denote the set of nodes as $\mathcal{A}$ instead of $\mathcal{A}_t(B)$ since we only consider a single node B that is not in the group and a particular group of nodes that can discover B. In a mobile network, as illustrated in Fig. 1(a), multiple instances of neighbor discovery in group settings can happen in parallel and are treated independently. The neighbor discovery schemes that we will develop (see Sections IV and V) are for general mobile network topologies, where nodes are duty cycled and do not have synchronized clocks; their

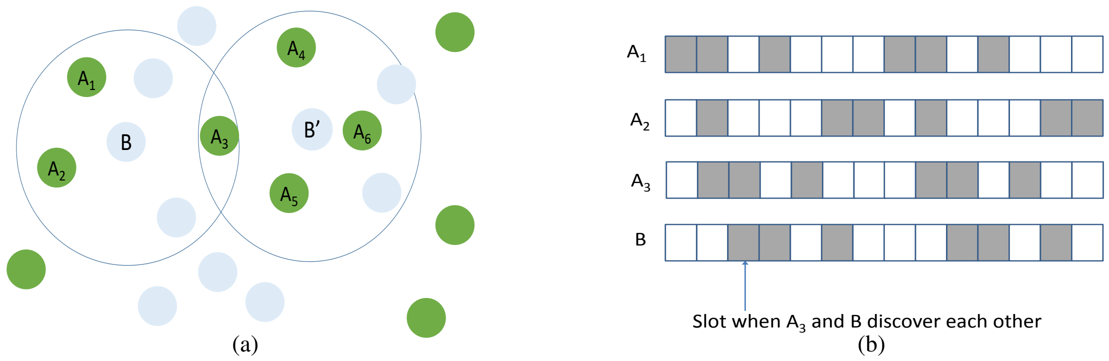


Fig. 1. (a) Illustration of neighbor discovery in group settings. (b) An example illustrating neighbor discovery in a group setting. The shaded slots represent awake slots.

applications in mobile networks will be demonstrated in a case study in Section VII.

For simplicity, we divide time into equal-length slots, labeled by $\{0, 1, \ldots\}$. A node is either awake or asleep in a time slot. When it is asleep, its radio is off and hence it cannot transmit or receive any message. When it is awake, the radio is on, and it can transmit, listen or alternate between transmitting and listening. Specifically, we assume that it transmits a beacon at the beginning and end of a slot, and listens for the rest of the slot. Using beacons at both ends of a slot is to accommodate slot misalignment, as in [10]. It permits the analysis to treat the schedules as though they are aligned at the slot level; for a detailed treatment of this alignment issue, see [6]. Two nodes, A and B, discover each other when they have an overlapping awake slot and successfully receive the beacons from each other.

Let S_i and d_i denote the schedule and duty cycle, respectively, of node $A_i \in \mathcal{A}$, where $S_i \subset \{0, 1, \ldots\}$ specifies the set of slots when A_i is awake. Likewise, let S and d denote the schedule and duty cycle of B. The goal is to determine the schedule for each node so that the latency for one node in $\mathcal{A}$ to discover B is minimized. Fig. 1(b) shows an example, where $\mathcal{A} = \{A_1, A_2, A_3\}$. The schedule of a node is represented by the slots, with shaded slots representing awake slots and blank slots representing asleep slots. The discovery latency in this example is 3 slots, when A_3 and B discover each other.

Existing pairwise schedules guarantee that a node $A_i \in \mathcal{A}$ can discover B within $c/(d_i \cdot d))$ slots, where c is a constant determined by the details of a pairwise discovery schedule (see Section II). For neighbor discovery in group settings, simply using an existing pairwise schedule, we can certainly guarantee discovery latency in time no more than

$$\frac{c}{(\max_i d_i) \cdot d}\,.$$

Note that this analysis just relies on the node with the largest duty from the group to discover B—in particular, it does not provide any benefit for a large group that consists of nodes with small duty cycles. Ideally, of course, we would like the nodes in $\mathcal{A}$ to work in tandem to provide stronger discovery guarantees that scale as a function of the *total duty cycle* of the group rather than the maximum duty cycle of the group.

We define the *effective duty cycle*, denoted d_{eff}, of the group as the fraction of time when at least one node in the group is awake. That is, d_{eff} is the duty cycle of the schedule $\bigcup_i S_i$. To calibrate our expectations, note that if each node A_i were awake with independent probability d_i in each time slot, then $\Pr[\text{no } A_i \text{ awake}] = \prod_i (1 - d_i)$ and hence $\Pr[\text{some } A_i \text{ awake}] = 1 - \prod(1 - d_i)$. With this motivation, we define the *ideal duty cycle*

$$d_{\text{ideal}} = 1 - \prod_i (1 - d_i) \approx 1 - \mathrm{e}^{-\sum_i d_i}\,,$$

where the approximation is accurate when the d_i's are small.

As an example, consider a collection of 10 nodes, each with 5% duty cycle. Then the ideal duty cycle of the group is $1 - (1 - 0.05)^{10} \approx 40\%$. When the nodes use pairwise schemes for neighbor discovery, since all the nodes have the same duty cycle, they may naively choose exactly the same schedule (i.e., they all wake up in exactly the same slots). In that case, the effective duty cycle of the group is only 5%, far below the ideal duty cycle.

Our goal shall be to provide schedules for which $d_{\text{eff}} \approx d_{\text{ideal}}$, and show that the nodes in a group can actually discover new nodes with latency determined by d_{eff} rather than $\max_i d_i$.

C. A Simple Randomized Schedule

We now show that a simple randomized schedule, where a node wakes up in each time slot with independent probability equal to its duty cycle, has the desired property that $d_{\text{eff}} \approx d_{\text{ideal}}$. For any fixed time slot, it is easy to show that for the randomized schedule $\Pr[\text{some } A_i \text{ is awake}] = 1 - \prod_i (1 - d_i)$, which is precisely d_{ideal} for this group. Then it is clear that

$$\Pr[\mathcal{A} \text{ and } B \text{ discover each other at time } t] = d_{\text{ideal}} d$$

for any particular time step t and, moreover,

$$\mathbb{E}[\text{discovery time for } \mathcal{A} \text{ and } B] = \frac{1}{d_{\text{ideal}} d}\,,$$

where $\mathbb{E}[\cdot]$ denotes expectation. It is furthermore easy to establish strong "tail bounds" on the probability that discovery takes significantly longer than T is:

$$\begin{aligned}\Pr[\text{discovery time for } \mathcal{A} \text{ and } B > T] &\le (1 - d_{\text{ideal}} d)^T \\ &\le \mathrm{e}^{-T d_{\text{ideal}} d}\,.\end{aligned}$$

Despite these strong properties, randomized schedules have a critical drawback: even after two nodes discover each other—due to the randomness in the schedules—they cannot predict

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3133099

each other's future awake slots. Additionally, randomized schedules cannot offer deterministic guarantees of discovery.

D. Deterministic Schedules

We focus on deterministic schedules, which have the advantage that when two nodes discover each other, they can precisely predict each other's awake slots in the future. This allows convenient future communications and can significantly improve application performance (see Section VII). We develop two deterministic schemes that provide $d_{\text{eff}} \approx d_{\text{ideal}}$ and also have the desirable property that two nodes can communicate their schedules to each other with short messages. The first scheme (Section IV) is a random shift technique that can be applied to existing pairwise discovery schedules, while maintaining their worst-case pairwise discovery guarantees. The second scheme (Section V) is a new family of "polynomial" schedules, which provides a rigorous guarantee on expected discovery time (scaling in d_{ideal}).

In both schemes, a node does not need to keep track of which group that it belongs to, or what other nodes are in the group, or communicate explicitly with other nodes in the group, hence they are ideal for mobile networks where the nodes in the group change dynamically over time. The "coordination" among the nodes in the group is through the randomness in their schedules, specifically, the random shift at the beginning of the schedule and the random coefficients of the polynomial for each node. Once the randomness is determined at the beginning, the schedules are deterministic. Both schemes lead to deterministic future awake times, and hence are deterministic schemes. The deterministic future awake times will be leveraged in Section VII to improve application-level performance. In addition, both schemes can be used in mobile networks with interference and packet collisions. In such cases, the beacons that are used for neighbor discovery can be corrupted, which can increase the discovery latency.

IV. Random Shift Schedules

Before describing the random shift technique, we first use an example to motivate its design. Suppose all nodes in $\mathcal{A}$ have the same duty cycle and the same deterministic schedule. In that case, they would all be awake during the same time slots, and thus effectively act like a single node. Therefore, having the group of nodes, no matter how large the group is, does not help at all in reducing the latency in discovering B. One simple approach to avoid the above synchronization is to add random shifts to the nodes' schedules, which is the main idea of the random shift technique. We next describe the approach, and then prove that it leads to an effective duty cycle that is close to the ideal duty cycle.

A. Schedule Design

Consider a group of nodes $\mathcal{A} = \{A_1, \ldots, A_k\}$, with schedules $S_1, \ldots, S_k$ and duty cycles $d_1, \ldots, d_k$, where S_i is determined by a pairwise scheme. The random shift technique can be applied to any deterministic pairwise schedules that are periodic, i.e., the schedule is a fix-length schedule that repeats itself. Specifically, denote the periods of the nodes' schedules as $n_1, \ldots, n_k$. The random shift schedules are determined by selecting a random shift α_i for each S_i, where α_i is selected independently and uniformly in $\{0, 1, \ldots, n_i - 1\}$; this choice suffices since schedule S_i is periodic, with period n_i.

Fig. 1(b) shows an example of random shift schedules. Suppose each node has duty cycle of $3/7$, and the original schedule is $\{0, 1, 3, 7, 8, 10, \ldots\}$, i.e., a periodic schedule with period of 7, which is determined by a pairwise neighbor discovery scheme. Following the random shift technique, each node chooses a random shift in $\{0, 1, \ldots, 6\}$ and applies it to the original schedule. Specifically, the random shifts for nodes A_1, A_2, A_3, and B are 0, 5, 1, and 2, respectively. Their schedules after the random shift are shown in Fig. 1(b), where the shaded slots represent awake slots. After the random shift, we see that the discovery latency is 3 slots, when A_3 discovers B. On the other hand, if nodes A_1, A_2 and A_3 all use the schedule as that of A_2, the discovery latency will be 6 slots.

As mentioned earlier, the above random shift technique can extend any existing periodic pairwise scheme to group settings. As an illustration, we apply it to Disco [10]. Recall that for a node with duty cycle d, the corresponding Disco schedule is characterized by two primes p_1 and p_2 so that $1/p_1 + 1/p_2 \approx d$. Applying the randomly shift technique, a node determines the two primes as above, and selects a uniformly random element from $\{0, 1, \ldots, p_1 p_2 - 1\}$. Note that with these parameters set, the schedule is now deterministic. When two nodes meet each other, they inform each other of the two primes as well as their current offset in the schedule; then each can completely reconstruct the other's future awake slots. The random shift does not introduce any additional communication overhead since even without random shift, a node needs to send another node its two primes and its current offset in the period (since nodes could start their schedules at different times).

B. Discovery Time

We next establish that, with high probability, random shift schedules achieve effective duty cycle d_{eff} that is close to d_{ideal}. Define $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$, where α_i is the random shift for node $A_i \in \mathcal{A}$. Then the effective duty cycle of the group $\mathcal{A}$ is

$$d_{\text{eff}}(\boldsymbol{\alpha}) \triangleq \frac{|\{t \in \{0, \ldots, \prod_i n_i - 1\} \mid \exists i, t + \alpha_i \in S_i\}|}{\prod_i n_i}, \quad (1)$$

where we use the notation $d_{\text{eff}}(\boldsymbol{\alpha})$ in place of d_{eff} as a reminder that it depends on $\boldsymbol{\alpha}$. Since the schedule of each node is periodic, when defining $d_{\text{eff}}(\boldsymbol{\alpha})$, it suffices to consider a period of $\prod_i n_i$, where n_i is the period of schedule S_i for node $A_i \in \mathcal{A}$.

Theorem 1. *Let $\mathcal{A} = \{A_1, \ldots, A_k\}$ be a group of nodes, with schedules $S_1, \ldots, S_k$, duty cycles $d_1, \ldots, d_k$ and periods $n_1, \ldots, n_k$. Under the random shift technique, the expected effective duty cycle $d_{\text{eff}}(\boldsymbol{\alpha})$ as defined in (1) satisfies*

$$\mathbb{E}[d_{\text{eff}}(\boldsymbol{\alpha})] = d_{\text{ideal}} = 1 - \prod_i (1 - d_i) \approx 1 - \mathrm{e}^{-\sum d_i}.$$

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3133099

In the "small" regime when $\sum_i d_i \leq 1/2$, for any $0 < \eta < 1$ we have

$$\Pr\left[d_{\mathrm{eff}}(\boldsymbol{\alpha}) < 1 - \mathrm{e}^{-(1-\eta)\sum_i d_i}\right] \leq \mathrm{e}^{-\frac{\eta^2 \sum_i d_i}{2\max_i d_i}}. \qquad (2)$$

It follows that $d_{\mathrm{eff}}(\boldsymbol{\alpha}) \approx d_{\mathrm{ideal}}$ with high probability, so long as $\max_i d_i \ll \sum_i d_i$.

Proof. Let D_i denote the density of the union of the first i shifted schedules $S_1+\alpha_1 \cup \cdots \cup S_i+\alpha_i$. Let $\Gamma_i = 1-D_i$ denote the density of the "uncovered" times when none of $A_1, \ldots, A_i$ are awake. With these definitions, note that $d_{\mathrm{eff}}(\boldsymbol{\alpha}) = D_k = 1 - \Gamma_k$.

Writing $\Gamma_i = (1-\gamma_i)\Gamma_{i-1}$ (for a value $0 \leq \gamma_i \leq 1$), the random variables Γ_i implicitly define a family of random variables

$$\gamma_i = \frac{\Gamma_{i-1} - \Gamma_i}{\Gamma_{i-1}}.$$

The variable γ_i is the fraction of "uncovered" times (by $A_1, \ldots, A_{i-1}$) that are newly covered by adding the schedule $S_i + \alpha_i$. A favorable property of γ_i is that, regardless of $S_1, \ldots, S_{i-1}$ and $\alpha_1, \ldots, \alpha_{i-1}$, the expected value of γ_i is precisely d_i (assuming that $\Gamma_i \neq 0$). (Recall that the probability that any individual time slot is covered by $S_i + \alpha_i$ is precisely d_i.) It follows that the random variables $X_k = \sum_{i=1}^k (\gamma_i - d_i)$ satisfy the martingale condition $\mathbb{E}[X_k \mid X_{k-1}] = X_{k-1}$. Note, also, that if $\sum_i d_i \leq 1/2$ then for each i we have $\Gamma_i \geq 1/2$ and $\Gamma_{i-1} - \Gamma_i \leq d_i$ so that $0 \leq \gamma_i \leq 2d_i$; it follows that $|X_i - X_{i-1}| \leq d_i$.

We apply now Azuma's inequality [25, Theorem 4.16], a classical concentration inequality for martingales. In this case, it asserts that

$$\Pr[X_k \leq -\lambda] \leq \mathrm{e}^{-\frac{\lambda^2}{2\sum_i d_i^2}}$$

and, by setting $\lambda = \eta \cdot \sum_i d_i$ for a constant $0 < \eta < 1$, that

$$\Pr\left[\sum_i^k \gamma_i \leq (1-\eta)\left(\sum_i^k d_i\right)\right] \leq \mathrm{e}^{-\frac{\eta^2(\sum_i d_i)^2}{2\sum_i d_i^2}} \leq \mathrm{e}^{-\frac{\eta^2 \sum_i d_i}{2\max_i d_i}} \qquad (3)$$

where we have applied the bound

$$\sum_i d_i^2 \leq \Big(\sum_i d_i\Big) \max_i d_i.$$

Observe, finally, that if $\sum \gamma_i \geq (1-\eta)\sum d_i$, then

$$\Gamma_k = \prod_i^k (1-\gamma_i) \leq \mathrm{e}^{-\sum_i \gamma_i} \leq \mathrm{e}^{-(1-\eta)\sum_i d_i}. \qquad (4)$$

Combining equations (3) and (4), we conclude that

$$\Pr[\Gamma_k > \mathrm{e}^{-(1-\eta)\sum_i d_i}] \leq \mathrm{e}^{-\frac{\eta^2 \sum_i d_i}{2\max_i d_i}}.$$

Recalling that $d_{\mathrm{eff}}(\boldsymbol{\alpha}) = 1-\Gamma_k$, this establishes (2) as desired. □

As an example, consider a group of 10 nodes and the duty cycle of each node is 1%. When setting $\eta = 1/2$, following (2), $d_{\mathrm{eff}}(\boldsymbol{\alpha})$ is at least 4.9% with 71.3% certainty. When the number of nodes is increased to 20, following (2), $d_{\mathrm{eff}}(\boldsymbol{\alpha})$ is at least 9.5% with 91.7% certainty. In both cases, the effective duty cycle is much larger than 1%, the duty cycle of an individual node.

Theorem 1 shows that the random shift scheme establishes control of d_{eff} for *arbitrary* schedules subjected to random shifts at the start of the schedules. Additionally, we remark that the construction retains any favorable pairwise discovery guarantees that the schedule might have previously enjoyed. Specifically, if the original schedules guaranteed pairwise discovery in time $c/(dd')$, then these new schedules also have this property. On the other hand, this result does not give a guarantee on the *expected* discovery time; the polynomial scheme (Section V) provides such a guarantee.

V. Polynomial Schedules

In this section, we develop a family of schedules that are determined by a small number of random bits (which can be thought of as the "fingerprint" of the schedules) and are otherwise deterministic; then other nodes which are aware of the "fingerprint" of a peer's schedule can completely reconstruct the peer's schedule. Furthermore, we wish the schedules to have guaranteed expected discovery time that scales appropriately in terms of d_{ideal}. To achieve such guarantees, we study a family of "polynomial" schedules that rely on degree-3 polynomials. The reason for focusing on degree-3 polynomials is that they achieve *4-wise independence*, a property critical for our analysis of the schedules (see Theorem 3).

A. Schedule Design

Consider a group of nodes $\mathcal{A}$ and node B. Each node $A_i \in \mathcal{A}$ has duty cycle d_i and schedule S_i. Node B has duty cycle d and schedule S. In the following, for ease of exposition, we first describe how B determines its schedule S based on degree-3 polynomials; the schedules for the nodes in $\mathcal{A}$ are constructed similarly (see later).

Schedule for B. Based on its duty cycle d, node B chooses a prime, $p > 1/d$. Let $\mathbb{Z}_p$ be the field with p elements, i.e., the integers modulo p. Define a polynomial $Q(x) = a_3x^3 + \cdots + a_0$, where $a_0, \ldots, a_3$ are chosen independently and uniformly in $\mathbb{Z}_p$. Then node B determines its schedule $S \subset \mathbb{Z}_p$ as

$$S = \{x \in \mathbb{Z}_p \mid Q(x) \bmod p \in \{0, \ldots, \lfloor pd \rfloor\}\}.$$

Since the coefficients $a_0, \ldots, a_3$ are chosen independently and uniformly in $\mathbb{Z}_p$, the p random variables

$$Q(0), Q(1), \ldots, Q(p-1)$$

are *4-wise independent* (see Remark below); in particular, for any 4 fixed, distinct elements $x_1, \ldots, x_4 \in \mathbb{Z}_p$, the random variables $Q(x_1), \ldots, Q(x_4)$ are uniform in $\mathbb{Z}_p$ and independent. This property will be critical for the proof of Theorem 3 below.

Remark. In general, we remark that if a degree k polynomial $P(x) = \sum_{i=0}^k b_i x^i$ is selected by choosing the coefficients $b_i \in \mathbb{Z}_p$ independently and uniformly at random, then for any $k+1$ distinct fixed values $v_0, \ldots, v_k$ the random variables $P(v_0), \ldots, P(v_k)$ are independent. To see this, consider a sequence of $k+1$ "target" values $w_0, \ldots, w_k \in \mathbb{Z}_p$ (which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3133099, IEEE Transactions on Wireless Communications

are not necessarily distinct) and the event that $\forall i, P(v_i) = w_i$. Organizing these constraints into a linear system, we find that the b_i must satisfy

$$\underbrace{\begin{pmatrix} 1 & v_0 & v_0^2 & \cdots & v_0^k \\ 1 & v_1 & v_1^2 & \cdots & v_1^k \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & v_k^1 & v_k^2 & \cdots & v_k^k \end{pmatrix}}_{V} \begin{pmatrix} b_0 \\ \vdots \\ b_k \end{pmatrix} = \begin{pmatrix} w_0 \\ \vdots \\ w_k \end{pmatrix}.$$

The matrix V in the above linear system is a Vandermonde matrix, and hence has full rank when the v_i are distinct. It follows that this linear system has a unique solution in the b_i and hence that $\Pr[\forall i, P(v_i) = w_i] = 1/p^{k+1}$, as desired. As this holds for any fixed collection of v_i, we conclude that the family of random variables $P(0), \ldots, P(p-1)$ are $k+1$-wise independent (when $k < p$).

We next describe two properties of the above schedule: (i) it preserves the duty cycle, and (ii) it leads to succinct representation. First of all, we observe that the above schedule preserves the duty cycle of the node. Specifically, for any fixed $x \in \mathbb{Z}_p$, $\Pr[Q(x) \in \{0, \ldots, \lfloor pd \rfloor\}] = (\lfloor pd \rfloor + 1)/p = d + r/p$, where $0 \leq r \leq 1$. Hence, the average duty cycle of the schedule S is

$$\mathbb{E}_{\mathbf{a}}[|S|/p] = (\lfloor pd \rfloor + 1)/p = d + O(1/p),$$

where $\mathbf{a} = (a_0, \ldots, a_3)$. We remark that the $O(1/p)$ error term can be made arbitrarily small by choosing a large value of p. In particular, for $p \approx 1000/d$, this error is never more than 0.1% and $\log_2 p \approx \log_2 1000 + \log_2 d \leq 10 + \log_2(1/d)$ so that integers modulo p can be efficiently written down and manipulated. We also emphasize that once $(a_0, \ldots, a_3)$ is chosen (at the beginning), the schedule is deterministic and can be represented succinctly through the prime p and the four values, $a_0, \ldots, a_3 \in \mathbb{Z}_p$. This succinct information can be forwarded to another node (after discovery), which can then completely reconstruct the schedule. The communication overhead for the setting discussed above (where $p = 1000/d$) is $6 \log_2 p \approx 60 + 6 \log 1/d$ bits.

Schedules for the nodes in $\mathcal{A}$. Following a similar procedure as above, each node $A_i \in \mathcal{A}$ chooses a prime based on its duty cycle d_i, and then chooses a random degree-3 polynomial, which determines its schedule S_i. For the schedules thus constructed, for any fixed time t it follows immediately that

$$\Pr\left[t \in \left(\bigcup_i S_i\right) \cap S\right] = d_{\text{ideal}} d \tag{5}$$

where $d_{\text{ideal}} = 1 - \prod(1 - d_i)$. That is, the probability for $\mathcal{A}$ to discover B at time t is $d_{\text{ideal}} d$. (The probability above is taken over the choices of the coefficient vectors for all the nodes in $\mathcal{A}$ and B.)

B. Discovery Time

Theorem 2. *Consider a group of nodes $\mathcal{A} = \{A_i\}$ and node B. Assume each node selects a schedule following the polynomial scheme as described above. Let S_i denote the schedule for node A_i, S denote the schedule for node B, and d_i (and d) the average duty cycle. Let X_t be the indicator variable for the event $t \in (\bigcup_i S_i) \cap S$. Then*

$$\mathbb{E}_{\boldsymbol{a}}[\textit{discovery time}] = \mathbb{E}_{\boldsymbol{a}}[\min\{t \mid X_t = 1\}] \leq 1 + \frac{5}{d_{\text{ideal}} d}, \tag{6}$$

where $d_{\text{ideal}} = 1 - \prod_i (1 - d_i)$.

The random variables X_t's as defined above are 4-wise independent (because the event X_t depends only on a single value $Q_i(x_i)$ for node i, where x_i is determined by the offset for this node). The above theorem is an immediate consequence of the following general result on "first nonzero" variable for a family of 4-wise independent variables.

Theorem 3. *Let $X_1, \ldots, X_T$ be a sequence of 4-wise independent indicator random variables with $\mathbb{E}[X_i] = \mu$. Then*

$$\mathbb{E}[\min\{t \mid X_t = 1\}] \leq 1 + 5/\mu.$$

We begin with a proof of Theorem 3 and then return to prove Theorem 2.

Proof of Theorem 3. We begin with a standard 4th moment inequality to bound the probability $\sum_i^t X_i = 0$ for $1 \leq t \leq T$. (The reader might wonder why a conventional second–moment method does not suffice for this purpose—we comment on this below.) For convenience we write $Y_i = X_i - \mu$, so that $\mathbb{E}[Y_i] = 0$ and note that

$$\begin{aligned} \mathbb{E}\left[\left(\sum_i^t Y_i\right)^4\right] &\leq \sum_{i_1, \ldots, i_4} \mathbb{E}[Y_{i_1} \ldots Y_{i_4}] \\ &= \sum_i \mathbb{E}[Y_i^4] + \binom{4}{2} \sum_{i<j} \mathbb{E}[Y_i^2 Y_j^2], \end{aligned}$$

as any term of the form $\mathbb{E}[Y_{i_1} \ldots Y_{i_4}]$ in which a particular index appears exactly once is equal to zero by virtue of the fact that the Y_i are independent and $\mathbb{E}[Y_i] = 0$. Observe then that $\mathbb{E}[Y_i^2 Y_j^2] = \mathbb{E}[Y_i^2]\,\mathbb{E}[Y_j^2]$ (when $i \neq j$) and $0 \leq \mathbb{E}[Y_i^4] \leq \mathbb{E}[Y_i^2] \leq \mu$, so that

$$\mathbb{E}\left[\left(\sum_i Y_i\right)^4\right] \leq \mu t + 3(\mu t)^2 \leq 4(\mu t)^2 \tag{7}$$

assuming that $t > 1/\mu$. Applying Markov's inequality, we conclude that (when $t \geq 1/\mu$)

$$\begin{aligned} \Pr\left[\sum X_i = 0\right] &\leq \Pr\left[\left|\sum Y_i\right| \geq t\mu\right] \\ &= \Pr\left[\left(\sum Y_i\right)^4 \geq (t\mu)^4\right] \\ &\leq \frac{\mathbb{E}\left[\left(\sum Y_i\right)^4\right]}{(\mu t)^4} \leq \frac{4}{(\mu t)^2}. \end{aligned}$$

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3133099, IEEE Transactions on Wireless Communications

It follows that the expected value of the smallest index i for which $X_i = 1$ is

$$\begin{aligned}
&\mathbb{E}\big[\min\{i \mid X_i = 1\}\big] \\
&= \sum_{t=1}^{T} \Pr[\min\{i \mid X_i = 1\} \geq t] \\
&= 1 + \sum_{t=1}^{T} \Pr[X_1 = \cdots = X_t = 0] \\
&\leq \sum_{t \leq \lceil 1/\mu \rceil} 1 + \sum_{t > \lceil 1/\mu \rceil} \Pr[X_1 = \cdots = X_t = 0] \\
&\leq \left\lceil \frac{1}{\mu} \right\rceil + \sum_{t > \lceil 1/\mu \rceil} \frac{4}{\mu^2 t^2} \\
&\leq \left\lceil \frac{1}{\mu} \right\rceil + \frac{4}{\mu^2} \int_{1/\mu}^{\infty} \frac{1}{t^2}\, dt \leq 1 + \frac{5}{\mu}\,.
\end{aligned}$$

We remark that a similar argument based only on variance (that is, the second moment) does not provide such a $O(\mu^{-1})$ bound on expectation, as the resulting integral $\int 1/t$ would introduce an undesirable factor of $\log T$ into the final bound. □

Remark 1. *A more careful analysis of the expectation above can reduce the factor 5 appearing in the denominator. Specifically, by separately handling the* (μt) *and* $(\mu t)^2$ *terms of* (7), *one can reduce the factor of* 5 *in the numerator of the final bound to* 4.5.

Proof of Theorem 2. Applying Theorem 3 to the X_i defined above (with expectation given by (5)) and $T \to \infty$ yields the desired bound (6). □

Last, the above polynomial algorithm is designed for neighbor discovery in group settings. In a pairwise setting, it does not guarantee that two nodes will discover each other in a bounded amount of time; to ensure delay guarantee, it can be used together with existing pairwise schemes as follows. For a duty cycle of d, select two duty cycles d_1, d_2 so that $d_1 + d_2 = d$, use the polynomial time algorithm to decide a wakeup schedule S_1 for duty cycle d_1, use an existing pairwise scheme to decide a wakeup schedule S_2 for duty cycle d_2, and then $S_1 \cup S_2$ is used as the wakeup schedule for duty cycle d.

VI. Experimental Results

In this section, we use experiments in a testbed to investigate the impact of various practical issues (e.g., interference, transmission latency) on neighbor discovery in group settings. We consider three schemes. The first two apply the random shift technique to two representative pairwise schemes: Disco [10] and Singer [40] (Singer is an optimal scheme for pairwise neighbor discovery in homogeneous settings; Disco works in both homogeneous and heterogeneous settings, and is significantly simpler than Singer), referred to as *Disco-RS* and *Singer-RS*, respectively. The third scheme is the polynomial scheme, referred to as *Poly*.

In the following, we first describe the testbed setting, and the experimental results on latency and loss rate under multiple simultaneous transmissions. After that, we describe the experimental results when running the neighbor discovery schemes in the testbed.

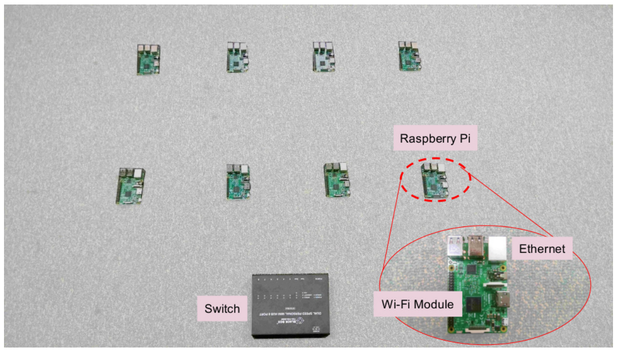
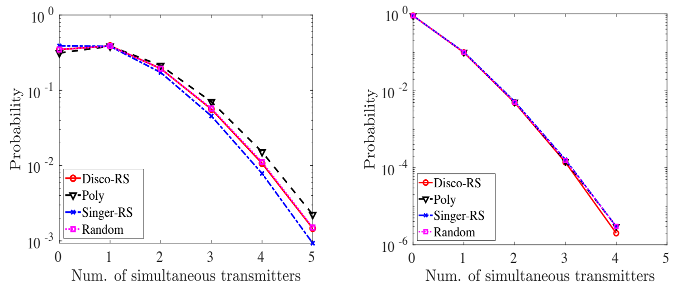


Fig. 2. Illustration of the testbed. Each Raspberry Pi has one WiFi and one Ethernet interface. The wireless interfaces are set in the ad-hoc mode for neighbor discovery. The Ethernet interfaces are connected to a switch (wires omitted for clarity) for easy control of the experiments.



(a) $d = 10\%$. (b) $d = 1\%$.

Fig. 3. Probability of having i ($i \geq 0$) simultaneous transmitters when $n = 10$ nodes are in the neighborhood of each other.

A. Testbed

Our testbed contains 8 Raspberry Pi 3 Model B devices (see Fig. 2) that are placed in the transmission range of each other. Each device has two network interface cards: an integrated BCM43438 WiFi interface that features single-band 2.4 GHz IEEE 802.11b/g/n, and a 100 Mbps Ethernet interface. The wireless interfaces of the devices are configured in the ad-hoc mode for neighbor discovery. They are set to communicate over channel 1, and use the IEEE 802.11 MAC protocol, i.e., CSMA/CA, for medium access. The beacons for neighbor discovery are sent using broadcast, and hence have no MAC-layer ACKs or retransmissions. The Ethernet interfaces are used to control the experiments. Specifically, we connect the Ethernet interfaces of the devices to a switch. The control commands related to the experiments are sent via the Ethernet interfaces through the switch so that the latency is negligible. One node serves as the *aggregator*, which sends the control commands to the other nodes, and collects the experimental results, all through the Ethernet network.

The operating system on each node is Raspbian 8. We program all the nodes using Python, leveraging the RPC/RMI (Remote Process Call/Remote Method Invocation) module Pyro (Version 4.60) [9], which allows us to define the aggregator and nodes conveniently as objects.

B. Simultaneous Transmissions

When multiple nodes transmit simultaneously (e.g., when their slots are aligned and they wake up in the same slot), they may potentially cause interference to each other. In the

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

following, we first quantify the extent of simultaneous transmissions in group settings (i.e., the probability that multiple nodes in the neighborhood transmit simultaneously), and then investigate its impact on transmission latency and packet loss rate using experiments in the testbed.

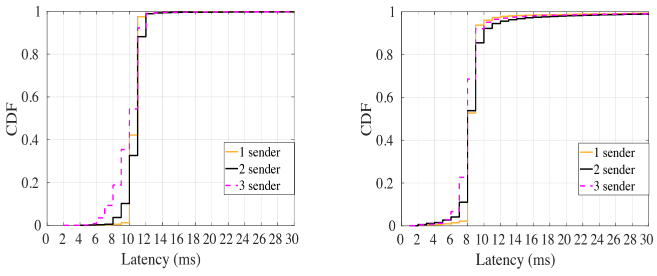
Extent of simultaneous transmissions. Suppose n nodes are in the transmission range of each other, each with duty cycle d. In the rest of the paper, we only consider the cases when n is around 10 or less, and the nodes' duty cycles are no more than 10%. The considered node density is reasonable since the transmission range of a node is within tens or hundreds of meters under existing technologies such as Zigbee and WiFi (note that n represents the number of nodes that are in the transmission range of each other; the total number of nodes in the network can be much larger, as in the settings to be explored in Section VII). Setting duty cycle to be no more than 10% is desirable for saving energy, particularly for embedded devices that are often used in wireless networks.

Fig. 3(a) plots the probability of having i nodes waking up and transmitting in the same slot under four neighbor discovery schemes (Disco-RS, Singer-RS, Poly, and the simple randomized schedule) when $n = 10$ and $d = 10\%$. The results for the simple randomized schedule are obtained analytically: under the scheme, since a node wakes up in a slot with probability equal to d, the probability of having i nodes waking up in the same slot is $\binom{n}{i}d^i(1-d)^{n-i}$. The results for the other three schemes are obtained empirically (from 10,000 instances of schedules). Fig. 3(b) plots the results under a lower duty cycle, $d = 1\%$. We observe that in both cases, the probability of having more than 3 simultaneous transmitters is very low: it is below 1.8% when $d = 10\%$ and below 0.03% when $d = 1\%$.

Experiment setup. We use experiments to investigate the impact of simultaneous transmissions on one-way latency and packet loss rate, which will be used to provide insights on the neighbor discovery results in Section VI-C.

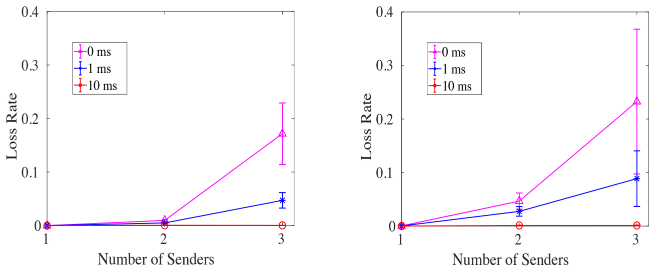
Based on the results above, we set the number of simultaneous transmitters to 2 or 3, and use the case when there is a single transmitter as the baseline. A node transmits UDP packets periodically every 100 ms over the ad-hoc wireless network. Each packet contains three fields, node ID, sending time and a sequence number. In addition, we set up a node as a sniffer, dedicated to listening and logging all the packets it hears. All the nodes are synchronized using Network Time Protocol (NTP) so that we can obtain one-way latency from sending to receiving a packet (we verified that the clock differences of the nodes are within 2 ms). The sniffer records the time when it receives a packet, and then obtains the one-way delay of the packet as the difference of the sending time (carried by the packet) and the receiving time. It further uses the sequence numbers to obtain packet loss rate.

The aggregator sends control messages to the nodes sequentially with an interval, Δ, through the Ethernet network to initialize the transmission of the nodes (i.e., a node immediately starts transmission after receiving a control message). In the following, we consider three settings, $\Delta = 0$, 1 or 10 ms. When $\Delta = 0$ ms, the aggregator sends control messages back-to-back to the nodes, essentially making the nodes transmit simultaneously; when $\Delta = 1$ or 10 ms, the transmissions of the nodes are staggered. Once starting, a transmitter sends 1200 UDP packets periodically at the interval of 100 ms. Our testbed shares the same wireless spectrum as the university campus WiFi network (our testbed is located in a lab inside the University of Connecticut). Each setting is repeated 5 times in midnight (when the load in the campus WiFi network tends to be low, and hence causing less interference to the testbed) and 5 times at noon time (when the load in the campus WiFi network tends to be higher).



(a) Midnight (b) Noon

Fig. 4. One-way latency from a sender to a receiver when the number of simultaneous senders is 1, 2, or 3, and $\Delta = 0$ ms.



(a) Midnight (b) Noon

Fig. 5. Packet loss rate when the number of simultaneous senders is 1, 2, or 3, and $\Delta = 0$, 1 or 10 ms.

Latency and loss rate. Fig. 4 plots the CDF (cumulative distribution function) of one-way delay when the number of transmitters is set to 1, 2, or 3, and $\Delta = 0$ ms (i.e., when the node transmissions are synchronized). The results for the measurements at midnight and noon time are both shown in the figure; the results for other settings (i.e., when Δ is 1 ms or 10 ms) are similar (figures omitted). We observe that one-way delay is around 10 ms in most cases. It is insensitive to the number of simultaneous transmitters. During noon time, the variance of the latency is slightly larger than that during midnight, indicating more impacts from the campus WiFi network.

Fig. 5 plots the average loss rate in each setting; the 95% confidence intervals are also plotted in the figure. In all the settings, we observe negligible loss rate when there is a single transmitter. When there are 2 or 3 transmitters, the loss rate is negligible when $\Delta = 10$ ms, which is not surprising since as shown in Fig. 4, most of the one-way latencies is within 10 ms, and hence staggering node transmissions by 10

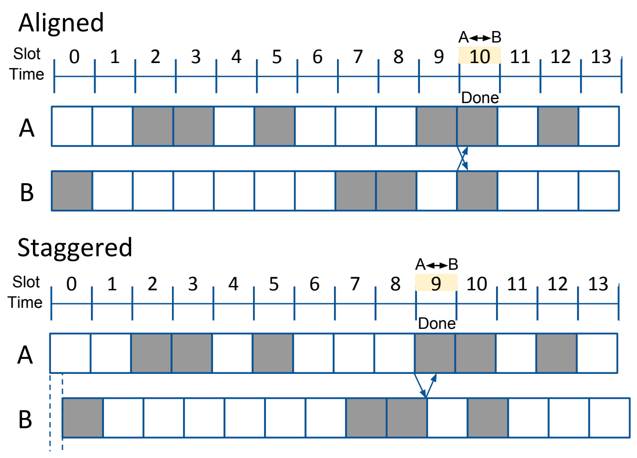


Fig. 6. Illustration of neighbor discovery with aligned and staggered slots. In this example, the discovery latency is 10 slots in the aligned case, and 9 slots in the staggered case.

ms causes very little interference to each other (and hence negligible loss rate). When $\Delta = 0$ ms, the loss rate can be high, particularly for 3 simultaneous transmitters. The loss rate is reduced significantly when the nodes' transmissions are slightly staggered (when $\Delta = 1$ ms). Last, comparing Figures 5(a) and (b), we observe slightly lower packet loss rates during midnight compared to that during noon time, which might be due to less background traffic during midnight in the campus WiFi network.

While the loss rate can be high when multiple nodes transmit simultaneously (i.e., when $\Delta = 0$ ms), it represents a scenario that is unlikely to occur during neighbor discovery in practice, since it requires the nodes to have perfectly synchronized clocks. As we have shown, even a slight offset of the nodes' transmissions leads to very low loss rate. In addition, as shown earlier, the probability of having more than two simultaneous transmitters is low under the settings we consider. Hence we do not expect packet loss to have significant impact on neighbor discovery, which we validate next by comparing results from the testbed (that incorporates realistic factors, including packet loss) and those from a simulator (that does not consider packet loss).

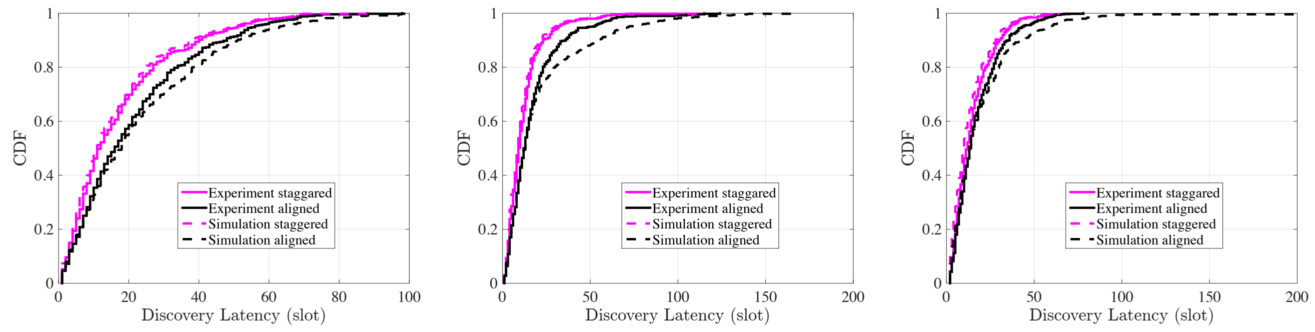
C. Results on Neighbor Discovery

We have implemented Disco-RS, Singer-RS, and Poly in the testbed. While running them in the testbed has the advantage of incorporating the impact of various realistic issues, it is cumbersome for evaluation in some cases (e.g., in mobile networks, or when the nodes duty cycles are very low and hence the discovery latency is very large). We therefore also implemented them in a simulator that we developed in MATLAB. The simulator captures the essential properties of the schemes, while making simplifying assumptions in that it does not consider transmission latency or packet loss. In the following, we compare the experimental results obtained from the testbed with those from simulations. In both the testbed and simulator, the number of nodes in $\mathcal{A}$ is set to 2, 4 or 6. All the nodes have the same duty cycle of 10%; the cases with lower and heterogeneous duty cycles are deferred to Section VII. We next describe the setups in the testbed and the simulator in more detail, and then the results.

In the testbed, a node follows a pre-calculated schedule: when it is asleep, it does not transmit or listen to packets; when it is awake, it broadcasts a beacon (a UDP packet with the node ID) at the beginning and end of a slot, and listens for the rest of the slot. The slot length is 100 ms or 50 ms (to be significantly larger than one-way delay, which is around 10 ms). We investigate two cases, i.e., when the slots of the nodes are aligned and when they are staggered, as illustrated in Fig. 6. Specifically, following the setup in Section VI-B, the aggregator sends control messages to the individual nodes so that their slots are aligned (i.e., $\Delta = 0$ ms), which can lead to high loss rate, or staggered by 10 ms (i.e., $\Delta = 10$ ms), which only leads to negligible loss rate. The nodes in $\mathcal{A}$ are started first, and B is started the last. When a node in $\mathcal{A}$ discovers B, it sends the discovery latency T (i.e., the time when the discovery happens minus the time when the neighbor discovery process starts) to the aggregator. Similarly, when B discovers a node in $\mathcal{A}$, it sends the discovery latency T' to the aggregator. The aggregator records the discovery latency in the group setting as $\max(T, T')$.

In the simulator, we model the neighbor discovery process to obtain the discovery latency. In the aligned case, the discovery is achieved when a node in $\mathcal{A}$ wakes up in the same slot as B, which is the standard notion of discovery in the literature. In the staggered case, due to the staggering and the one-way latency, discovery can be achieved when B is in an awake slot t and a node $A \in \mathcal{A}$ is in an awake slot $t+1$. Therefore, under the same schedules, the staggered case can achieve faster discovery than the aligned case. Fig. 6 illustrates these two cases using an example: in the aligned case, the discovery is achieved in slot 10 for both nodes A and B; in the staggered case, the discovery is achieved in slot 8 for B and slot 9 for A.

Fig. 7(a)-(c) show the CDF of the discovery latency of Singer-RS, Disco-RS and Poly, respectively. Each distribution is obtained from 500 runs, with randomly chosen schedules in each run, when each node has duty cycle of 10% and $|\mathcal{A}| = 6$. The slot length is 100 ms (using slot length of 50 ms leads to similar results). The results from both the testbed and simulator are shown in the figure. We observe that the staggered case indeed leads to faster discovery, and the results from the testbed match well with those from the simulator. For the aligned case, the latency from the testbed tends to be smaller than that from the simulator. This may be because of stochastic transmission latencies and/or less aligned slots over time (e.g., due to different speeds of the clocks) in the testbed. Overall, for all the three schemes, the results obtained from the simulator agree with those from the testbed. Last, we remark that the one-way latency (i.e., around 10 ms) on our Raspberry Pi platform is much larger than that in other platforms (e.g., Mote sensors [5], [10], programming radios [23], low-power RFID platforms [7], [21], and other embedded platforms [17], [26]). The slot length of the neighbor discovery schemes running on a platform needs to be chosen based on one-way latency of the platform. Existing studies used slot length of 10 ms or less for the platforms that they use; we used larger slot length due to much larger one-way latency of our platform.



(a) Singer-RS. (b) Disco-RS. (c) Poly.

Fig. 7. Experimental and simulation results for Singer-RS, Disco-RS and Poly. Each node has duty cycle of 10%, $|\mathcal{A}| = 6$.

VII. Case Study

In this section, we present a case study that uses neighbor discovery in group settings as a building block. Specifically, we consider two applications described in Section III-A: (i) unicast data transfer in a DTN network, where a source needs to send a message to a destination and multiple copies of the message (each carried by an individual node) are allowed in the network; and (ii) broadcast data transfer where the message needs to be sent to the rest of the nodes in the network. For the unicast application, the goal is to minimize *delivery latency*, i.e., the delay from the message is generated at the source until it reaches the destination; for the broadcast application, the goal is to minimize *completion time*, i.e., the delay from the message is generated at the source until it reaches all the other nodes. For our proposed random shift and polynomial techniques, we develop a simple extension called *Proactive Transmission* that leverages the deterministic nature of these schedules and show that it can significantly improve application-level performance (i.e., reducing delivery latency or completion time). In the following, we first describe the Proactive Transmission mechanism, and then the simulation setting and results.

A. Proactive Transmission

The Proactive Transmission (PT) mechanism is used by a node that has *just* received a copy of the message to *proactively* transmit it to other nodes that might be close by and have not received a copy of the message. It allows the message to be transmitted faster into the network, and hence reducing the delivery latency.

Specifically, suppose that a node A has just received a copy of the message at time t. Suppose that node A maintains a *node list*, N_A, which records a list of nodes in the network that A is aware of (because node A met them in the past or through message exchange with other nodes, see later). For each node $u \in N_A$, let $u.r = 1$ represent that, to node A's knowledge, u has received a copy of the message; $u.r = 0$ otherwise. Let $u.t$ represent the time when node A last met u and $u.\alpha$ represent the set of parameters of u that can be used to determine u's future awake slots. Through the PT mechanism, node A sends a copy of the message to up to K nodes in N_A. Specifically, node A considers each node $u \in N_A$ with $u.r = 0$ (i.e., A believes that u has not received a copy of the message), sorts them in increasing order of $t - u.t$ (breaking ties arbitrarily), and selects the first (up to) K nodes, i.e., preferring the nodes that A met more recently. For each selected node v, node A determines the next time slot when v is awake (based on v's schedule, as represented in $v.\alpha$), denoted as t_v, and proactively wakes up at t_v to transmit a copy of the message to v. Then v will receive a copy of the message if and only if it is in node A's transmission range at time t_v. As described above, for a particular message, node A only uses the PT mechanism once (i.e., when it just receives a copy the message), which leads to low overhead.

In the PT mechanism above, we assume that a node does not know its location (e.g., the nodes are simple embedded devices that do not have GPS). Therefore, a node simply uses the last meeting time as a proxy of its distance from another node. When nodes are equipped with GPS and node distance can be estimated more accurately, a more refined mechanism can be developed for A to decide which K nodes to select to increase the likelihood of success.

Maintaining node list. We now describe how a node, A, maintains its node list, N_A. Initially, N_A is empty. When node A meets another node B at time t, they update their status information (i.e., whether one has received a copy of the message or not). That is, if A has a copy of the message, while B does not, then A transmits a copy of the message to B, and vice versa. Then node A updates its node list N_A: it adds B to N_A (if $B \notin N_A$), with B's schedule $B.\alpha$, the latest meeting time $B.t = t$, and the status information $B.r$. Node B updates its node list N_B in a similar way. In addition, A and B share their node lists so that they get to know more nodes in the network as follows. Consider each node $u \in N_A \cup N_B$. If $u \in N_A$ and $u \in N_B$, then A and B update u's status based on their joint information: if either A or B knows that u has received a copy of the message, then both of them set $u.r = 1$ in their node lists. If $u \in N_B$ but $u \notin N_A$, then A adds u to its node list N_A, with u's information, $u.r$ (as recorded by B), $u.\alpha$, and sets $u.t = -\infty$ (indicating that A has not met u yet). Similarly, if $u \in N_A$ but $u \notin N_B$, then B adds u to its node list N_B in a similar manner.

B. Simulation Setting

We use a custom-built simulator that we developed in MATLAB to simulate the unicast and broadcast applications in

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3133099

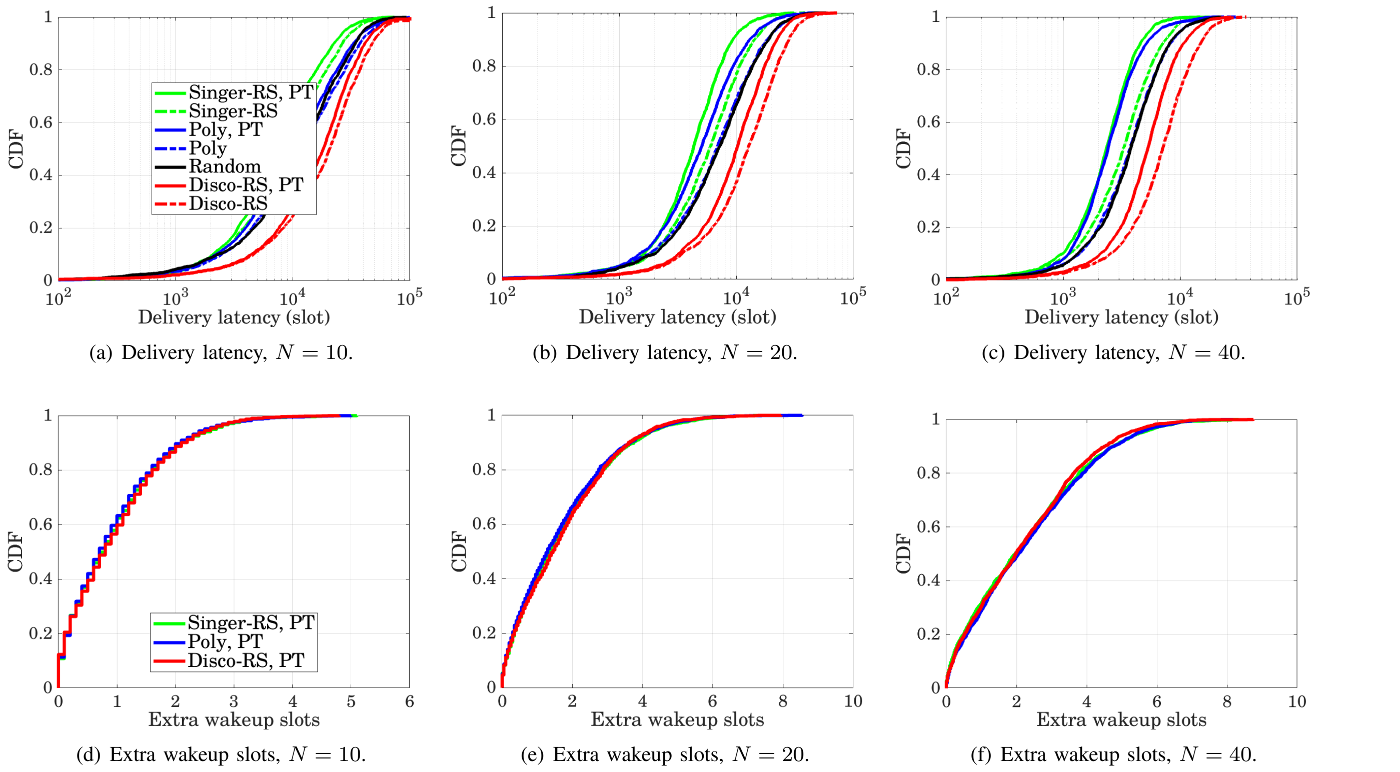


Fig. 8. Performance results for the unicast application in homogeneous setting (all the nodes have duty cycle of 1%), $p_s = 1$, and $K = 10$ in PT.

a mobile network. In the network, N nodes move in an area of 500m × 500m following Random Waypoint Model. The simulation of neighbor discovery follows the same methodology as described in Section VI-C. Following the literature, we assume the slots of the nodes are aligned, and hence two nodes discover each other when they are both in awake slots. The slot length for neighbor discovery is set to 10 ms. For simplicity, we assume each node has a circular transmission range of radius $R = 150$ m. We set N to 10, 20, or 40; correspondingly, the average number of neighbors for a node is 2.9, 5.4, and 11.0. For all the nodes, the minimum and maximum moving speeds are 0.5 m/s and 2 m/s, respectively. One node, chosen randomly from the N nodes, is the original data source that has a message. In the unicast application, the message needs to be sent to a destination that is chosen randomly. In the broadcast application, the message needs to be sent to all the other nodes in the network. For both applications, when the source discovers a node u, it transmits a copy of the message to u, which then becomes an additional data source and can transmit the message to other nodes. As time goes on, more and more nodes have a copy of the message. For the unicast application, the process ends when the destination receives a copy of the message. For the broadcast application, we explore two cases: the process stops after all the nodes receive the message or a finite amount of time.

At one point of time, as illustrated in Fig. 1(b), there can be multiple instances of neighbor discovery in group settings, one for each node B that has not yet received a copy of the message, and a group of nodes that are in B's transmission range and each has received a copy of the message. A scheme that leads to faster neighbor discovery in group settings allows B to get a copy of the message more quickly, and reduces the overall delivery latency to the destination(s). We consider four schemes: Disco-RS, Singer-RS, Poly, and the simple randomized algorithm serving as the baseline. For the first three schemes, we consider the variants with and without PT. The randomized algorithm cannot perform PT due to the randomized nature of its schedules.

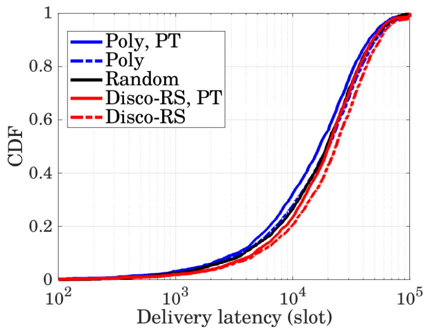
To model the impact of interference and packet collision or loss, we assume that when two nodes are in the transmission range of each other and are both awake, they can discover each other successfully with probably $p_s \in (0, 1]$. In the following, we set p_s to 1 (ideal case), 0.9, or 0.5 (high interference).

The performance metrics are (i) latency, i.e., delivery latency for the unicast application and completion time for the broadcast application, and (ii) the average number of extra wakeup slots per node in a simulation run, which is only applicable to the three deterministic schemes with PT. For each setting, the results below are obtained from 2,000 simulation runs, each lasting up to 10^5 time slots.

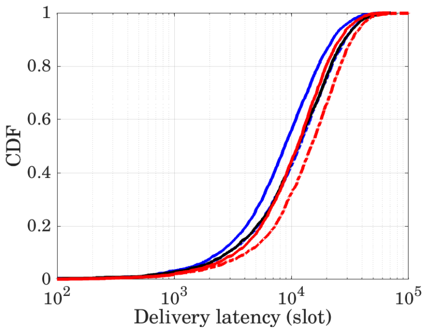
C. Evaluation Results for the Unicast Application

In the following, unless otherwise stated, $p_s = 1$ (i.e., two nodes discover each other when they are in the transmission range of each other and both are awake). At the end of this subsection, we explore the impact of p_s.

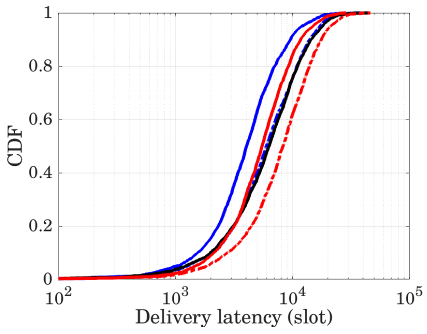
Homogeneous settings. Fig. 8(a)-(c) plot the CDF of the delivery latency for the various schemes when all the nodes have duty cycle $d = 1\%$, $p_s = 1$, $K = 10$ in PT, and the number of nodes N in the network is set to 10, 20, and 40, respectively. (i) We first compare Singer-RS, Disco-RS and Poly without PT with the randomized algorithm. Among the four schemes,



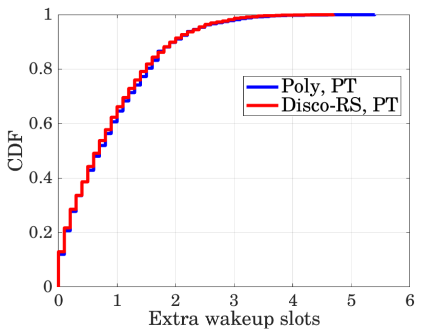
(a) Delivery latency, $N = 10$.



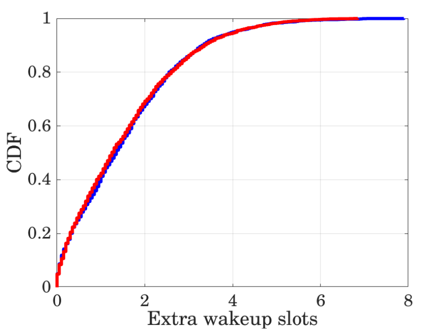
(b) Delivery latency, $N = 20$.



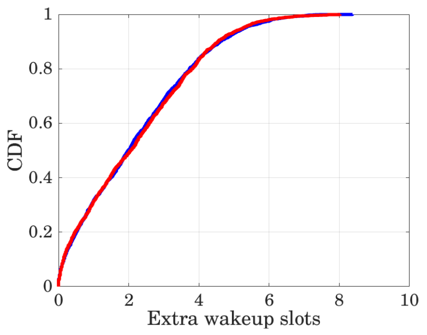
(c) Delivery latency, $N = 40$.



(d) Extra wakeup slots, $N = 10$.



(e) Extra wakeup slots, $N = 20$.



(f) Extra wakeup slots, $N = 40$.

Fig. 9. Performance results for the unicast application in heterogeneous settings (nodes randomly choose their duty cycles to be either 0.5% or 1%) , $p_s = 1$, and $K = 10$ in PT.

Singer-RS leads to the shortest delivery latency for all the settings, particularly when the network density is low (i.e., $N = 10$). This is not surprising, since it is based on Singer, an optimal pairwise discovery scheme for homogeneous settings, which provides advantages particularly in early stage (when less nodes have a copy of the message in the network). The performance of Poly is similar to that of the randomized algorithm, particularly for $N = 20$ and 40, where they almost overlap with each other. The performance of Disco-RS lags behind the other three algorithms. (ii) When adding PT, the performance of all the three deterministic schemes (Singer-RS, Disco-RS, Poly) improves significantly. Both Singer-RS with PT and Poly with PT have visibly shorter delivery latency than the randomized algorithm for all settings of N. When $N = 10$, their median delivery latencies are 40% and 20% lower than that of the randomized algorithm, respectively; the corresponding reductions are respectively 40% and 30% when $N = 20$, and 38% and 36% when $N = 40$. Disco-RS with PT leads to delivery latency significantly closer to that of the randomized algorithm for larger N (i.e., $N = 20$ and 40), compared to Disco-RS without PT.

Fig. 8(d)-(f) plot the CDF of the average number of extra wakeup slots per node in a simulation run. The results for the three schemes with PT are plotted in the figure. Since $K = 10$, a node has at most 10 extra wakeup slots in an entire simulation run. We see that the extra overhead due to PT is similar for the three schemes, and is low in all the settings. The extra overhead increases with the number of nodes N in the network, with the average number of extra wakeup slots per node less than 6 in most of the simulation runs even when $N = 40$.

Heterogeneous settings. We assume the original data source has duty cycle of 1%, while for the rest of the nodes, each node randomly chooses its duty cycle to be either 0.5% or 1%. Again $K = 10$ in PT, $p_s = 1$, and the node density is varied by setting $N = 10$, 20 or 40. Fig. 9(a)-(c) plot the delivery latency of the various schemes (Singer based schemes are not shown in the figures since they are only applicable to homogeneous settings). Without PT, we again observe that the delivery latency of Poly is similar to that of the randomized algorithm. When adding PT to Poly, its delivery latency becomes significantly lower than that of the randomized algorithm, particularly for large N: its median delivery latency is 13%, 23%, and 36% lower than that of the randomized algorithm when N is 10, 20 and 40, respectively. The delivery latency of Disco-RS without PT is worse than that of the randomized algorithm; Disco-RS with PT has similar or slightly lower delivery latency compared to that of the randomized algorithm. Fig. 9(d)-(f) again shows that PT only leads to a small number of extra wakeup slots per node.

Impact of p_s. So far, our results are for the ideal settings of $p_s = 1$. When $p_s < 1$, the neighbor discovery latency will be larger, which will lead to larger delivery latency. Fig. 10(a) plots the CDF of the delivery latency in homogeneous settings, when $N = 40$ and $p_s = 0.9$. As expected, for all the schemes, the delivery latency is approximately 10% higher than that when $p_s = 1$ (shown in Fig. 8(c)).

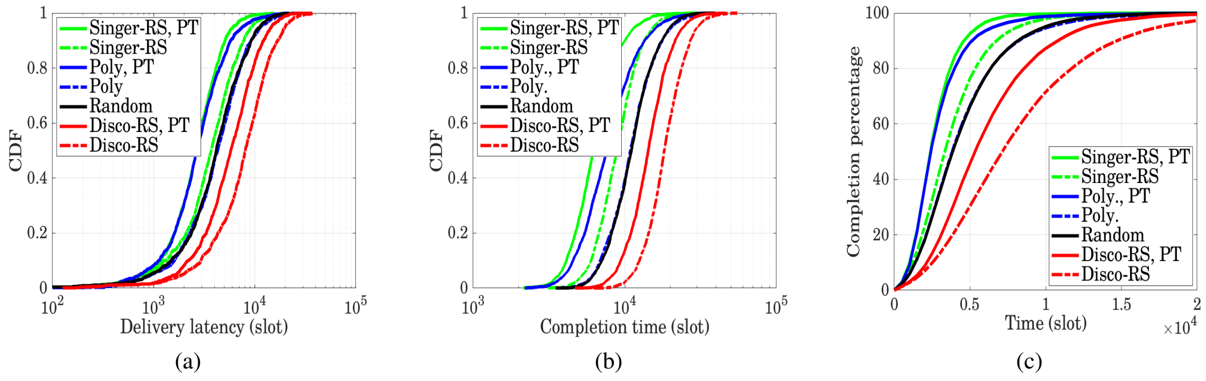


Fig. 10. (a) Impact of p_s: delivery latency for the unicast application, homogeneous setting when $p_s = 0.9$, $N = 40$, and $K = 10$ in PT. (b) and (c) are for the broadcast application, homogeneous setting when $p_s = 1$, $N = 40$, and $K = 10$ in PT.

D. Evaluation Results for the Broadcast Application

We observe similar trends in the broadcast application as the unicast application. One example is shown in Fig. 10(b), which plots the CDF of the completion time for the homogeneous setting where all the nodes have duty cycle of 1%, and $N = 40$, $p_s = 1$, and $K = 10$ for PT. As in the unicast application, we see that adding PT significantly improves the performance of the deterministic schemes, and Singer-RS and Poly with PT significantly outperform the simple randomized algorithm.

We further explore the scenario where the message delivery process stops at time T. This is an interesting scenario since sometimes there is no need for other nodes to receive the message beyond a certain time, e.g., when the messages are generated periodically and a message becomes stale after time T. In this case, we explore the *completion percentage*, i.e., the percentage of the nodes that have received the message by time T. Fig. 10(c) plots the completion percentage when T is varied from 0 to 2×10^4 slots for the setting in Fig. 10(b). When $T = 10^4$ slots, the percentage of completion varies from 71.5% to 99.6% for the various schemes, while the percentage of runs that are completed by T (i.e., the runs in which all the nodes have received the message by T) is much lower (1.2% to 89.9% as shown in Fig. 10(b)), indicating that the majority of the nodes may receive the message quickly, while it may take a long time for all the nodes to receive the message.

VIII. Conclusion and Future Work

In this paper, we have investigated neighbor discovery in group settings. We first defined the notion of ideal duty cycle for a group, and then developed two deterministic algorithms, one based on random shifts and the other based on 3-degree random polynomials. We showed that for both schemes, the effective duty cycle of a group is close to the ideal duty cycle, and the polynomial schedule further provides a guarantee on expected discovery latency. In addition, being deterministic algorithms, they can be enhanced with a proactive transmission mechanism that leads to improved performance at little additional overhead, as shown in our case study.

As future work, we plan to improve the energy consumption of the neighbor discovery schemes as follows. Currently, the duty cycles of the nodes are fixed. As the group size increases (i.e., more nodes receive a copy of a message), it may be beneficial to reduce the duty cycle of the nodes, which can further preserve the energy of the nodes, while may not increase the application-level latency much. Exploring this direction in both unicast and broadcast applications is left as future work.

References

[1] M. Bakht, M. Trower, and R. H. Kravets. Searchlight: Won't you be my neighbor? In *Proc. of ACM MobiCom*, 2012.
[2] S. A. Borbash, A. Ephremides, and M. J. McGlynn. An asynchronous neighbor discovery algorithm for wireless sensor networks. *Ad Hoc Networks*, 5(7):998–1016, 2007.
[3] D. Burghal, A. S. Tehrani, and A. F. Molisch. On expected neighbor discovery time with prior information: Modeling, bounds and optimization. *IEEE Transactions on Wireless Communications*, 17(1), January 2018.
[4] L. Chen, R. Fan, K. Bian, M. Gerla, T. Wang, and X. Li. On heterogeneous neighbor discovery in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2015.
[5] L. Chen, Y. Shu, Y. Gu, S. Guo, T. He, F. Zhang, and J. Chen. Group-based neighbor discovery in low-duty-cycle mobile sensor networks. *IEEE Transactions on Mobile Computing*, 15(8), August 2016.
[6] S. Chen, R. Morillo, Y. Qin, A. Russell, R. Jin, B. Wang, and S. Vasudevan. Asynchronous neighbor discovery on duty-cycled mobile devices: Models and schedules. *IEEE Transactions on Wireless Communications*, 19(8), August 2020.
[7] T. Chen, J. Ghaderi, D. Rubenstein, and G. Zussman. Maximizing broadcast throughput under ultra-low-power constraints. In *Proc. of CoNext*, 2016.
[8] R. Cohen and B. Kapchits. Continuous neighbor discovery in asynchronous sensor networks. *IEEE/ACM Transactions on Networking*, 19(1), 2011.
[9] I. de Jong. Pyro: Python remote objects. https://pythonhosted.org/Pyro.
[10] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proc. of ACM SenSys*, 2008.
[11] D. Gao, Z. Li, Y. Liu, and T. He. Neighbor discovery based on cross-technology communication for mobile applications. *IEEE Transactions on Vehicular Technology*, 69(10):11179–11191, October 2020.
[12] C. S. Hsu, J. R. Jiang, Y. C. Tseng, and T. H. Lai. Quorum-based asynchronous power-saving protocols for ieee 802.11 and hoc networks. *ACM Journal on Mobile Networks and Applications (MONET)*, 10(1-2):169–181, 2005.
[13] J.-H. Huang, S. Amjad, and S. Mishra. CenWits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proc. of ACM SenSys*, 2005.
[14] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of SIGCOMM*, 2004.
[15] L. Jiang, J.-H. Huang, A. Kamthe, T. Liu, I. Freeman, J. Ledbetter, S. Mishra, R. Han, and A. Cerpa. SenSearch: GPS and witness assisted tracking for delay tolerant sensor networks. In *Proc. of International Conference on Ad-hoc and Wireless Networks (ADHOC-NOW)*, 2009.

[16] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[17] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar. U-Connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proc. of IPSN*, 2010.

[18] S. Lai, B. Zhang, B. Ravindran, and H. Cho. CQS-Pair: Cyclic quorum system pair for wakeup scheduling in wireless sensor networks. In *Proc. of Principles of Distributed Systems*, 2008.

[19] D. Li and P. Sinha. RBTP: low-power mobile discovery protocol through recursive binary time partitioning. *IEEE Transactions on Mobile Computing*, 13(2):263–273, 2014.

[20] C. Liu, G. Zhang, W. Guo, and R. He. Kalman prediction-based neighbor discovery and its effect on routing protocol in vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(1), January 2020.

[21] R. Margolies, G. Grebla, T. Chen, D. Rubenstein, and G. Zussman. Panda: Neighbor discovery on a power harvesting budget. In *Proc. of IEEE INFOCOM*, 2016.

[22] M. J. McGlynn and S. A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proc. of ACM MobiHoc*, October 2001.

[23] T. Meng, F. Wu, and G. Chen. On designing neighbor discovery protocols: A code-based approach. In *Proc. of IEEE INFOCOM*, 2014.

[24] T. Meng, F. Wu, and G. Chen. Code-based neighbor discovery protocols in mobile wireless networks. *IEEE/ACM Transactions on Networking*, 24(2):806–819, April 2016.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[26] A. Purohit, N. Priyantha, and J. Liu. WiFlock: collaborative group discovery and maintenance in mobile sensor networks. In *Proc. of IPSN*, 2011.

[27] Y. Qiu, S. Li, X. Xu, and Z. Li. Talk more listen less: Energy-efficient neighbor discovery in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2016.

[28] A. Russell, S. Vasudevan, W. Zeng, X. Chen, B. Wang, and W. Wei. Neighbor discovery in wireless networks with multipacket reception. *IEEE Transactions of Parallel and Distributed Systems*, 26(7):1984–1998, July 2015.

[29] W. Sun, Z. Yang, K. Wang, and Y. Liu. Hello: A generic flexible protocol for neighbor discovery. In *Proc. of IEEE INFOCOM*, 2014.

[30] W. Sun, Z. Yang, X. Zhang, and Y. Liu. Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 16(3):1448–1459, 2014.

[31] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proc. of IEEE INFOCOM*, 2002.

[32] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley. Efficient algorithms for neighbor discovery in wireless networks. *IEEE/ACM Transactions on Networking*, 21(1), February 2013.

[33] K. Wang, X. Mao, and Y. Liu. Blinddate: A neighbor discovery protocol. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):949–959, 2015.

[34] Y. Wang, G. Sun, G. Yang, and X. Ding. XgBoosted neighbor referring in low-duty-cycle wireless sensor networks. *IEEE Internet of Things Journal*, 8(5):3446–3461, March 2021.

[35] F. Wu, T. Meng, A. Li, G. Chen, and N. H. Vaidya. Have you recorded my voice: Toward robust neighbor discovery in mobile wireless networks. *IEEE/ACM Transactions on Networking*, 26(3):1432–1445, June 2018.

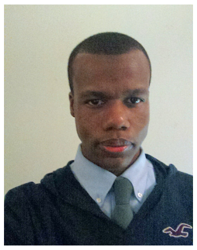
[36] D. Zhang, T. He, Y. Liu, Y. Gu, F. Ye, R. K. Ganti, and H. Lei. Acc: generic on-demand accelerations for neighbor discovery in mobile applications. In *Proc. ACM SenSys*, 2012.

[37] D. Zhang, T. He, F. Ye, R. K. Ganti, and H. Lei. EQS: neighbor discovery and rendezvous maintenance with extended quorum system for mobile sensing applications. In *Proc. of IEEE ICDCS*, 2011.

[38] L. Zhang and D. Guo. Neighbor discovery in wireless networks using compressed sensing with Reed-Muller codes. In *Proc. of WiOpt*, 2011.

[39] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *Proc. of ACM SenSys*, 2004.

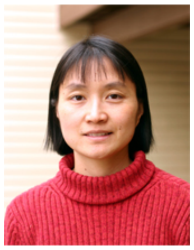
[40] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proc. of ACM MobiHoc*, 2003.



Reynaldo Morillo received a B.S. degree in Computer Science and Engineering from the University of Connecticut in 2015. He is currently pursuing his doctoral degree in the Computer Science & Engineering Department at the University of Connecticut. His research interests are in Computer Networks and Distributed Systems, and Network Security.

Yanyuan Qin received a B.S. degree in automation from Nanjing University of Aeronautics and Astronautics, China in 2011 and an M.S. degree in control science and engineering from Shanghai Jiao Tong University, China in 2014. He received a Ph.D. in Computer Science from the University of Connecticut in 2020. His research interests include wireless networking and software defined networking.

Alexander Russell holds a B.A. from Cornell University (1991), and both an S.M. (1993) and Ph.D. (1996) from the Massachusetts Institute of Technology. He is currently a professor of Computer Science at the University of Connecticut.



Bing Wang is currently a Professor of the Computer Science & Engineering Department at the University of Connecticut. She received her B.S. degree in Computer Science from Nanjing University of Science & Technology, China in 1994, and M.S. degree in Computer Engineering from Institute of Computing Technology, Chinese Academy of Sciences in 1997. She then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 2000, 2004, and 2005 respectively. Her research interests are in Computer Networks, Multimedia, and Distributed Systems. She received NSF CAREER award in 2008.