U-Connect: A Low-Latency Energy-Efficient Asynchronous Neighbor Discovery Protocol

Arvind Kandhalu, Karthik Lakshmanan, Ragunathan (Raj) Rajkumar
Real-Time and Multimedia Systems Laboratory
Carnegie Mellon University, Pittsburgh, PA
{akandhal, klakshma, raj}@ece.cmu.edu

ABSTRACT

Mobile sensor nodes can be used for a wide variety of applications such as social networks and location tracking. An important requirement for all such applications is that the mobile nodes need to actively discover their neighbors with minimal energy and latency. Nodes in mobile networks are not necessarily synchronized with each other, making the neighbor discovery problem all the more challenging. In this paper, we propose a neighbor discovery protocol called *U-Connect*, which achieves neighbor discovery at minimal and predictable energy costs while allowing nodes to pick dissimilar duty-cycles. We provide a theoretical formulation of this asynchronous neighbor discovery problem, and evaluate it using the *power-latency product* metric. We analytically establish that U-Connect is an 1.5-approximation algorithm for the symmetric asynchronous neighbor discovery problem, whereas existing protocols like Quorum and Disco are 2-approximation algorithms. We evaluate the performance of U-Connect and compare the performance of U-Connect with that of existing neighbor discovery protocols. We have implemented U-Connect on our custom portable *FireFly Badge* hardware platform. A key aspect of our implementation is that it uses a slot duration of only 250μs, and achieves orders of magnitude lower latency for a given duty cycle compared to existing schemes for wireless sensor networks. We provide experimental results from our implementation on a network of around 20 sensor nodes. Finally, we also describe a *Friend-Finder* application that uses the neighbor discovery service provided by U-Connect.

Categories and Subject Descriptors

C.2.1 [**Computer-Communications Networks**]: Network Architecture and Design–Wireless Communication

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IPSN'10, April 12–16, 2010, Stockholm, Sweden.
Copyright 2010 ACM 978-1-60558-988-6/10/04 ...$10.00.

General Terms

Algorithms, Design, Wireless Sensor Networks

Keywords

Mobile, Neighbor Discovery, Wireless, Sensor Networks

1. INTRODUCTION

Electronic devices have become increasingly more sophisticated and miniaturized. This has resulted in a natural shift in technology towards making these devices portable i.e. mobile. In the recent past, considerable interest has been shown towards mobile sensor networking by the wireless sensor networking community. Mobile sensor nodes have been used for a wide variety of applications such as detecting zebra-to-zebra interactions [15], social networking [18], logging hiker sightings after their trail-side encounters with other hikers [11] and biker experience mapping [8]. In all such applications, the mobile nodes need to actively discover their emerging mobile or static neighboring nodes, along with the attrition of old neighbors while operating their radios at low duty cycles to maximize their lifetime. The fundamental problem in neighbor discovery is that energy consumption and discovery latency are inversely related. That is, low duty-cycling of radios will result in lower energy consumption but higher latency of transmission/reception, and vice versa. Hence, an optimal trade-off solution for neighbor discovery is desirable. The *power-latency product* metric effectively models this requirement. We use this metric to determine the performance of a given neighbor discovery protocol. The properties of *power-latency product* are two-fold:

1. For any given latency requirement, a protocol with a lower *power-latency product* will always achieve lower power operation compared to any other protocol with higher *power-latency product*.
2. For any given power consumption, a protocol with a lower *power-latency product* will always achieve lower worst-case neighbor discovery latency than any other protocol with higher *power-latency product*.

1.1 Contributions

The main contributions of this paper are the following:

- Theoretical formulation of the asynchronous neighbor discovery problem,

- Analysis of existing neighbor discovery protocols [23, 7] using the *power-latency product* and establishing that they are 2-approximation algorithms,
- Development of a novel asynchronous neighbor discovery protocol called *U-Connect*,
- An analytical proof that *U-Connect* is an 1.5 approximation algorithm for the symmetric asynchronous neighbor discovery problem,
- An efficient implementation that uses a slot size of 250μs which provides for very-low discovery latencies for a given duty-cycle,
- Evaluation of the performance of U-Connect under both symmetric and asymmetric operation,
- Illustration of the U-Connect protocol in operation using a *Friend Finder* application.

In U-Connect, the communication schedule of each mobile node is divided into well-defined time slots. The schedule of the radio-listen pattern follows the low-power listening (LPL) scheme [9, 19] where the radio listens for transmissions on a periodic basis. The period between consecutive listen-slots is determined by the desired duty-cycle. An additional constraint in our protocol is that the period value needs to be prime in the number of slots. Such wake-ups of the node at multiples of prime numbers ensures deterministic discovery latency which is an outcome of the Chinese Remainder Theorem [12]. The transmission schedule consists of the nodes transmitting for half the listen period every hyper-cycle, the hyper-cycle being the square of the listen period. We call this transmission schedule pattern as *low-power transmit* (LPT).

We have also developed a mobile sensor node platform called *"FireFly Badge"*. The name comes from the fact that these devices will be placed into personnel badges and credit cards. The FireFly Badge is basically a mobile version of the FireFly sensor networking hardware platform. We have implemented U-Connect onto the Badges and evaluate its performance. We also compare the performance of U-Connect against many of the existing neighbor discovery protocols.

1.2 Organization

The structure of the paper is as follows. We discuss related work in Section II. In Section III, we provide a theoretical formulation of the neighbor discovery problem and analyze the performance of existing protocols using the power-latency metric. In Section IV, we describe the functioning of our protocol U-Connect and provide a theoretical analysis of the performance of U-Connect. In Section V, we present a detailed evaluation of U-Connect. In Section VI, we describe our design of the FireFly badge hardware platform and implementation of U-Connect. Section VII provides a summary of our experimental results and describes the Friend-Finder application that employs U-Connect. Finally, in Section VIII, we summarize our contributions and provide concluding remarks.

2. RELATED WORK

When a non-power-constrained node has knowledge regarding the physical presence of its neighbors in its communication range, a simple broadcast will result in the discovery of the nodes. Under power constraints and with a general consensus of the operating duty-cycle, discovery can be achieved with predictable latency using protocols that employ LPL with fixed listen periods such as BMAC [19], SMAC [24] and Quorum-based protocols [23, 14]. In Quorum, time is divided into a sequence of beacon intervals which are grouped into sets of m^2 contiguous intervals, where m is a global parameter that depends on the required duty-cycle. The m^2 intervals or slots are arranged as an $m \times m$ row major matrix. Each node picks a row and a column, and is awake during the slots represented by the corresponding values. When two nodes each pick an arbitrary row and column, they will have two intersections. These intersections imply that both the nodes will be awake during those 2 slots, resulting in the mutual discovery of the nodes. It is worth commenting that B-MAC and S-MAC implicitly use a slightly modified Quorum-based neighbor discovery protocol. The neighbor discovery problem becomes more complex in energy-constrained, mobile environments since the presence of a neighbor cannot be guaranteed during the scheduled wake up slots of the node. Additionally, the operating duty-cycles of nodes might be different depending upon their energy-latency requirements. The behavior of nodes operating at different duty-cycles with each node picking its duty-cycle independently with no clock synchronization is known as an *asynchronous asymmetric* operation. Tseng et al. [23] have proposed a Quorum-based protocol for multi-hop ad hoc networks. As mentioned earlier, *m* is a global parameter. This implies that all nodes operate at the same duty-cycle. Such an operating condition is known as a *symmetric* operation.

McGlynn and Borbash have proposed "birthday protocols" for asynchronous neighbor discovery in static ad hoc networks [17]. They considered the problems of energy conservation during node deployment and energy-efficient neighbor discovery following deployment using a scheme in which nodes listen, transmit or sleep with different probabilities. Their work concludes that mobile ad hoc networks would not use discovery, but that discovery would be useful in static ad hoc networks. Even in static networks, however, they schedule an explicit discovery phase that quickly terminated. The probabilistic discovery leads to aperiodic and unpredictable discovery latencies, and long tails on discovery probabilities, reducing its appeal. This has been noted in [7], and therefore, we do not compare against this protocol.

Zheng et al. [25] apply optimal block designs using difference sets to the problem of symmetric neighbor discovery. They suggest the use of the Multiplier Theorem [3] to obtain the optimal block designs. They have also looked into the problem of asymmetric duty cycling of static nodes in a network for optimal trade-off between energy and latency.

The authors conclude that their approach reduces to an NP-complete minimum vertex-cover problem requiring a centralized solution. We add that the use of difference sets towards solving asynchronous asymmetric duty-cycled neighbor discovery has not been explored in detail yet.

Dutta and Culler [7] have proposed a protocol named "Disco" for asynchronous neighbor discovery. It is an adaptation of the Chinese Remainder Theorem, where the nodes select a pair of prime numbers such that the sum of their reciprocals is equal to the desired duty cycles. The nodes then wakeup at multiples of the individual prime numbers. The worst-case discovery latency is the product of the minimum of the primes picked by the nodes in operation. The authors suggest the use of *balanced* primes (the difference between the primes of the individual node is minimal) for symmetric discoveries, and *unbalanced* primes (where the difference is maximal) for asymmetric discoveries. This behavior provides for deterministic discovery latencies that are much better than Quorum and birthday protocols for asymmetric scenarios, and similar to Quorum in the symmetric case. The inherent problem in such a protocol is that the choice of the primes have to be tuned for either asymmetric or symmetric scenario but not both. If the nodes that come in the vicinity of each other have picked the same unbalanced primes, the worst-case latency will then be very large.

The popular Bluetooth technology [1] also does neighbor discovery in order to achieve *pairing*. Once pairing is achieved, communication takes place freely between the paired devices. It uses frequency-hopping over 79 1MHz channels and has a bandwidth of 1Mbps. The specification of the Bluetooth technology is targeted at a different application scenario compared to the applications targeted by us. Bluetooth applications are usually meant to replace cables and are data-intensive. The Bluetooth protocol is too processor-intensive and consumes too much space for extremely resource-constrained mobile sensor networking platforms.

3. ASYNCHRONOUS NEIGHBOR DISCOVERY

In this section, we introduce the asynchronous neighbor discovery problem, provide its corresponding theoretical formulation, and develop relevant evaluation metrics for mobile sensor networks. For the benefit of the reader, we first introduce the symmetric neighbor discovery problem, and later generalize the formulation to describe the asymmetric neighbor discovery problem.

Mobile nodes in sensor networks continually move in space. Due to such motion, the network topology changes over time, nodes within communication range may move farther apart, and nodes outside the communication range move closer to each other. The neighbor discovery problem is one in which each mobile node keeps track of all other nodes within its communication range. This information changes over time and a neighbor discovery protocol is used to continuously update it. The asynchronous neighbor discovery problem is often encountered in mobile networks, where the mobile nodes discovering each other are not necessarily synchronized with each other.

The practical limitations on hardware clocks and synchronization algorithms in the mobile context restrict the level of synchronization achievable between mobile nodes. Therefore, we restrict our attention to slotted asynchronous neighbor discovery protocols, where the communication schedule of each mobile node is divided into well-defined slots. Our theoretical formulation of the asynchronous neighbor discovery protocol follows.

3.1 Theoretical Formulation

We first define the notion of a discovery schedule, which determines the intervals of time at which each mobile node performs neighbor discovery.

Discovery Schedule: The discovery schedule of a mobile node m is defined as the binary function $\Psi(m,t)$, which determines whether the node m attempts neighbor discovery at time instant t. $\Psi(m,t) = 0$ implies that the mobile node m is not in discovery mode at time instant t. t is relative to the time instant at which both the nodes come within communication range of each other. Conversely, $\Psi(m,t) = 1$ denotes that the mobile node m is in discovery mode at time instant t.

Next, we define the notion of neighbor discovery. A neighbor discovery between nodes m_1 and m_2 is defined to have happened at time instant t, if m_1 and m_2 are within the communication range of each other at time instant t, and $\Psi(m_1,t) = \Psi(m_2,t) = 1$.

For a given set of n nodes, the neighbor discovery problem involves the construction of $\Psi(m,t)$ for each mobile node $1 \leq m \leq n$. The construction should ensure that all pairs of nodes $(i,j), i \neq j$, discover each other, whenever they are within each other's communication range for a duration greater than or equal to L units of time. L is defined as the latency requirement for neighbor discovery. A solution to the neighbor discovery problem is a discovery schedule that satisfies the condition:

$$\forall i,j \mid 1 \leq i,j \leq n \; \exists \; 0 \leq t \leq L$$
$$such\,that\; \Psi(i,t) = \Psi(j,t) = 1 \qquad (1)$$

For simplicity of introduction, we begin our discussion with the synchronous symmetric neighbor discovery problem.

Synchronous Symmetric Neighbor Discovery Problem: The synchronous symmetric neighbor discovery problem with latency requirement L is defined as the neighbor discovery problem where all the n mobile nodes in the network have been assigned the same discovery schedule and the same origin of time i.e. $\Psi(i,t) = \Psi(j,t)\ \forall 1 \leq i,j \leq n\ \forall t$, and the maximum discovery latency is required to be less than or equal to L. In practice, not all mobile nodes in the system can have the same notion of time due to the difficulty of

achieving synchronization. The neighbor discovery problem is challenging when nodes have slightly different clock drifts and highly varying phase offsets. As described later, the clock drifts are handled through time-slotting mechanisms. Therefore, we restrict our attention to mobile nodes with a relative phase offset between them. The asynchronous symmetric neighbor discovery problem in this context is defined as follows.

Asynchronous Symmetric Neighbor Discovery Problem: The asynchronous symmetric neighbor discovery problem with latency requirement L is defined as the neighbor discovery problem where all the n mobile nodes in the network have been locally assigned the same discovery schedule and relative phase offsets i.e. $\Psi(i,t) = \Psi(j, t + \Phi_{i,j})\ \forall 1 \leq i, j \leq n$ $\forall t$ where $\Phi_{i,j}$ is the relative phase offset between the origin of time in nodes i and j, and the maximum discovery latency is required to be less than or equal to L.

3.2 Evaluation Metrics

We are primarily concerned with the problem of asynchronous neighbor discovery in the context of mobile sensor networks. Energy efficiency and low discovery latency are the key metrics involved in evaluating asynchronous symmetric neighbor discovery protocols in this context. There exists a trade-off between achieving lower energy operation and reducing the discovery latency. In order to perform a holistic evaluation of the neighbor discovery protocols, we consider the power-latency (PL) product Λ as the performance metric. We define the PL product Λ as the product of the average power consumption with the worst-case neighbor discovery latency in an ideal communication channel.

A periodic neighbor discovery schedule $\Psi(m,t)$ with period T is one which satisfies the condition $\Psi(m,t) = \Psi(m, t+T)\ \forall\ t$. The average power consumption of this schedule is given by:

$$P = \tfrac{1}{T} \int_0^T \Psi(m,t)\, dt$$

For the given periodic neighbor discovery schedule Ψ, let the worst-case discovery latency among all pairs of nodes with all possible relative phase offsets be L. The *power-latency* product Λ is defined as:

$$\Lambda = PL = \tfrac{L}{T} \int_0^T \Psi(m,t)\, dt$$

3.3 Slotted Discovery Schedules

Practical neighbor discovery algorithms such as those described in [23,25,17,7] use slotted discovery schedules. The advantages of slotted discovery mechanisms are that (i) they are practical to implement, and (ii) time-slotting mechanisms can overcome clock drift as long as the total drift is less than the duration of a single slot. Given a latency requirement of L, the slot duration should be chosen to be greater than the worst-case relative time drift over a duration of L time units, among all pairs of nodes in the system. The slot duration also depends on other implementation considerations such as the radio turn-around time to switch between Receive and Transmit, time to spin up the oscillator from low-power states, and clear channel assessment requirements. Given the slotted nature of the neighbor discovery algorithms in practice, it is useful to restrict our attention to discovery schedules that are discrete in nature. Let ψ denote the discretized version of Ψ. For periodic schedules ψ with period T satisfying latency requirement L, the power required is:

$$P = \tfrac{1}{T} \sum_{t=0}^{T-1} \psi(m,t)$$

and, the PL product Λ is:

$$\Lambda = \tfrac{L}{T} \sum_{t=0}^{T-1} \psi(m,t)$$

3.4 Analysis of Existing Neighbor Discover Schedules

We now analyze the performance of existing neighbor discovery protocols using the *power-latency product* metric. We use the $[x]_y$ notation to denote the modulus of an integer x with respect to y.

First, we introduce the theoretically optimal neighbor discovery schedules proposed in [25]. These schedules are not easy to generate for arbitrary latency requirements and are not amenable to asymmetric duty cycles, which are considered in this paper. Using these optimal schedules as the theoretical reference point, we demonstrate that existing neighbor discovery algorithms like Quorum [23] and Disco [7] are 2-approximation algorithms of the optimal. However, the U-Connect protocol achieves a 1.5-approximation of the optimal, and also solves the asymmetric asynchronous neighbor discovery problem. We have used approximation algorithms for performance comparison of different neighbor discovery algorithms as it is a well known technique applied to minimization problems in computer science. For the convenience of the reader we state the definition of an approximation algrithm here: An algorithm A is said to be an α-approximation [6] for a minimization problem β, *iff* for all instances of β, A guarantees a solution no more than αS where S is the optimal (minimum) solution.

3.4.1 Optimal

Theoretically optimal neighbor discovery schedules were discussed in [25]. It was shown that optimal schedules exist for all discovery latencies $L = k^2 + k + 1$, where k is any power of any prime. For these optimal schedules ψ_o, the total power P_o required is known to be (for details, please see [25]):

$$P_o = \frac{k+1}{k^2+k+1} = \frac{\sqrt{L - \frac{3}{4}} + \frac{1}{2}}{L} \qquad (2)$$

The PL product for the optimal schedule Λ_o is:

$$\Lambda_o = \sqrt{L - \frac{3}{4}} + \frac{1}{2} \tag{3}$$

Generating ψ_o for arbitrary latency requirements is not a well-understood problem. The scheme is also not readily amenable to asymmetric duty cycles, as was shown in [25]. In this paper, we are concerned with a unified asynchronous neighbor discovery protocol, which handles both symmetric and asymmetric duty cycles. Therefore, for our discussions, the optimal design serves more as a theoretical reference point to compare the performance of other algorithms.

3.4.2 Quorum

The asynchronous symmetric neighbor discovery problem has been addressed earlier by the Quorum protocol [23]. For simplicity of presentation, we assume that the latency requirement L is a perfect square. The Quorum protocol is illustrated in Figure 1(a). Given a latency requirement of L, the Quorum protocol proposes a class of neighbor discovery schedules ψ_q as follows:

$$\psi_q(m,t) = \begin{cases} 1, & \text{if } [t]_{\sqrt{L}} = c, \text{ or } r \leq \frac{[t]_L}{\sqrt{L}} < (r+1) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where, $0 \leq c, r < (\sqrt{L})$.

The discovery schedule is illustrated in the form of a $\sqrt{L} \times \sqrt{L}$ matrix shown in Figure 1(a). The chosen column for discovery is c, and the chosen row for discovery is r. The entire schedule repeats periodically every L units of time. Hence, $T = L$, for Quorum. It has been proven in [23] that the worst-case discovery latency is L. The PL product for Quorum Λ_q is:

$$\Lambda_q = \frac{L}{T} \sum_{t=0}^{T-1} \psi_q(m,t) = \frac{L}{L} \sum_{t=0}^{L-1} \psi_q(m,t) = (2\sqrt{L} - 1) \tag{5}$$

The quorum protocol is extensible to handle asymmetric duty cycles and rectangular discovery schedules, as shown in [5]. The discovery latency between asymmetrically duty-cycled nodes will be analyzed later. As can be seen from Equations (3) and Equations (5), asymptotically, the quorum protocol is a 2-approximation algorithm for the symmetric asynchronous neighbor discovery problem (for non-asymptotic performance see Figure 5). The motivation behind considering such an asymptotic behavior is that the energy budgets are expected to be really low in mobile wireless sensor devices, resulting in a relatively long worst-case discovery latency L.

$$\lim_{L\to\infty} \frac{\Lambda_q}{\Lambda_o} = \lim_{L\to\infty} \frac{2\sqrt{L} - 1}{\sqrt{L - \frac{3}{4}} + \frac{1}{2}} = 2 \tag{6}$$

3.4.3 Disco

The Disco protocol [7] has been proposed for the asynchronous neighbor discovery problem. The protocol works for both symmetric and asymmetric duty cycles. We first focus on the symmetric analysis and will study its asymmetric behavior later. Under the Disco protocol, a pair of primes is chosen to determine the neighbor discovery schedule. Whenever the slot number modulo any of the two primes is zero, nodes attempt a neighbor discovery. This scheme is illustrated in Figure 1(b). The class of neighbor discovery schedules proposed by Disco ψ_d is given by:

$$\psi_d(m,t) = \begin{cases} 1, & \text{if } [t]_{p_1} = 0 \text{ or } [t]_{p_2} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where p_1 and p_2 are $primes$. For the symmetric neighbor discovery problem, the worst-case discovery latency occurs when both the nodes choose the same pair of primes (p_1, p_2). The worst-case latency in this scenario is given by:

$$L_d = L = p_1 p_2 \tag{8}$$

The schedule is periodic with a period of $T = p_1 p_2$. Therefore, the power consumption in this scenario is given by:

$$P_d = \frac{1}{T} \sum_{t=0}^{T-1} \psi_d(m,t) = \frac{1}{p_1 p_2} \sum_{t=0}^{p_1 p_2 - 1} \psi_d(m,t) = \frac{p_1 + p_2}{p_1 p_2} \tag{9}$$

The PL product for Disco is therefore:

$$\Lambda_d = L_d P_d = p_1 + p_2 \tag{10}$$

$$\geq 2\sqrt{p_1 p_2} = 2\sqrt{L} \tag{11}$$

(since arithmetic mean $\geq$ geometric mean). Therefore, Λ_d for Disco is at least $2\sqrt{L}$.

As can be seen from Equations (3) and (11), asymptotically, the Disco protocol is at most a 2-approximation algorithm for the symmetric asynchronous neighbor discovery problem (for non-asymptotic behavior see Figure 5).

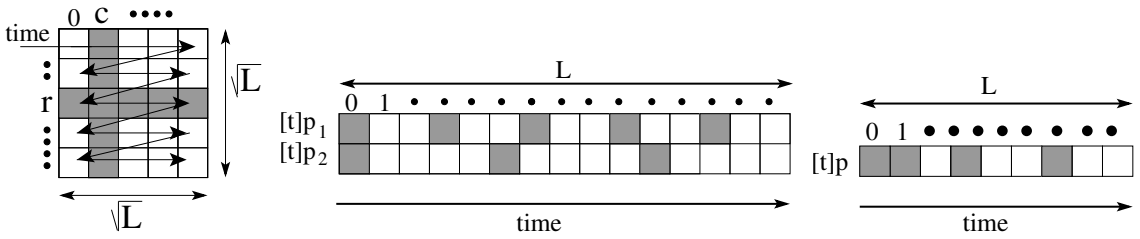
$$\lim_{L\to\infty} \frac{\Lambda_d}{\Lambda_o} \geq \lim_{L\to\infty} \frac{2\sqrt{L}}{\sqrt{L - \frac{3}{4}} + \frac{1}{2}} = 2 \tag{12}$$

As we will demonstrate in Section IV, our U-Connect protocol is a 1.5-approximation algorithm for the symmetric asynchronous neighbor discovery problem, and also performs much better on asymmetric duty cycles. Next, we introduce the U-Connect neighbor discovery protocol.

4. U-CONNECT PROTOCOL DESIGN

In this section, we develop the design of U-Connect, characterize its latency, analyze its power consumption, and evaluate it using the *power-latency* metric.

U-Connect is an unified protocol to address both the symmetric and asymmetric asynchronous neighbor discovery problems. Consider a protocol in which a node picks a prime number p and stays active for 1 every p slots. Any two primes are by definition relatively prime with each other. Hence,



(a) Quorum Illustration (b) Disco Illustration (c) Our Protocol Illustration

Figure 1: Illustration of the communication schedule under different protocols

two nodes choosing different primes will discover one another. If the nodes choose the same pair of primes, discovery can still be ensured by each node staying active for slightly longer than half the prime period p. Utilizing the theoretical formulation developed in the previous section, U-Connect is described as the class of neighbor discovery schedules ψ_u given by:

$$\psi_u(m,t) = \begin{cases} 1, & \text{if } [t]_p = 0 \text{ or } 0 \leq [t]_{p^2} < \frac{p+1}{2} \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

where p is a $prime$. For simplicity of presentation, we consider the case where $p > 2$ (a special case analysis can be performed for $p = 2$.) The U-Connect protocol is shown in Figure 1(c) which illustrates the motivation behind the design of the protocol.

4.1 Symmetric Asynchronous Neighbor Discovery

In the context of symmetric asynchronous neighbor discovery, all the nodes in the system have the same neighbor discovery schedules, and hence the same prime value for the parameter p.

Lemma 1: The worst-case discovery latency under U-Connect for a neighbor discovery schedule ψ_u with parameter p, where p is a prime, is less than or equal to p^2.

Please refer to [13] for the proof.

The U-Connect protocol is periodic in nature with a period $T = p^2$. The power consumption of U-Connect is, therefore, given by:

$$P_u = \frac{1}{p^2} \sum_{t=0}^{p^2-1} \psi_u(m,t) = \frac{\frac{p+1}{2} + p}{p^2} = \frac{3p+1}{2p^2} \quad (14)$$

The PL metric Λ_u is therefore:

$$\Lambda_u = \frac{p^2}{p^2} \sum_{t=0}^{p^2-1} \psi_u(m,t) = \frac{3p+1}{2} = \frac{3\sqrt{L}+1}{2} \quad (15)$$

As can be seen from Equations (3) and Equations (15), asymptotically, the U-Connect protocol is a 1.5-approximation algorithm for the symmetric asynchronous neighbor discovery problem (for non-asymptotic behavior refer to Figure 5).

$$\lim_{L\to\infty} \frac{\Lambda_u}{\Lambda_o} = \lim_{L\to\infty} \frac{\frac{3\sqrt{L}+1}{2}}{\sqrt{L - \frac{3}{4}} + \frac{1}{2}} = \frac{3}{2} \quad (16)$$

4.2 Average-Case Performance

We developed the worst-case latency for U-Connect, and evaluated its performance using the *power-latency* metric. In order to better understand the behavior of U-Connect, it would be useful to obtain a characterization of its average-case performance. We now provide the cumulative distribution function (c.d.f) for the discovery latency under U-Connect for the symmetric asynchronous neighbor discovery problem. For an overview of the average-case performance of the other neighbor-discovery protocols, please see [7].

Consider two nodes a and b with a relative offset of $\Phi_{a,b}$. Let their schedules be decided by the prime p. For simplicity of presentation, we develop the case where $p > 2$. A special-case analysis can be done for $p = 2$. Also, let the nodes encounter each other at a particular time instant t. Due to space constraints, we provide an asymptotic analysis here, which holds good for large p values. The asymptotic analysis captures the overall behavior.

For large p and uniformly distributed $\Phi_{a,b}$, we can ignore the cases where $[\Phi_{a,b}]_{p^2} < p$, since these happen with an arbitrarily low (for large values of p) probability of $\frac{1}{p}$. The cases with $[\Phi_{a,b}]_p = 0$ and $[\Phi_{a,b}]_p = \frac{p-1}{2}$ can also be ignored for the same reason. For other values of $\Phi_{a,b}$, there is only a single slot s every p^2 units of time, during which nodes a and b are simultaneously in discovery mode. Assuming the time instant t is chosen in an uniformly random fashion, the c.d.f of the time to discovery l can be written as:

$$Pr(l < x) = Pr([s-t]_{p^2} < x) = \frac{x}{p^2} = \frac{x}{L}.$$

The c.d.f of the asymptotic discovery latency for U-Connect is therefore a straight line with slope $\frac{1}{p^2} = \frac{1}{L}$. It can be seen that at $x = L$, the $Pr(l < L) = 1$ showing that the worst-case discovery latency for U-Connect is L. The mean time to discovery μ_l under asymptotic conditions is then given by:

$$\mu_l = \sum_{x=0}^{x<L} \frac{x}{L} = \frac{L-1}{2}$$

4.3 Asymmetric Operation

Asymmetric duty cycles are required in many applications that make use of neighbor discovery protocols. For instance, mobile nodes are more resource-constrained and hence require lower duty cycles, whereas powered nodes in the infrastructure can have arbitrarily high duty cycles. End users

can also configure their mobile devices to have different duty cycles depending on their latency requirements and battery-life expectations. A unified solution to the neighbor discovery problem should also cater to the asynchronous asymmetric neighbor discovery problem.

Asynchronous Asymmetric Neighbor Discovery Problem: The asynchronous asymmetric neighbor discovery problem with latency requirement L is defined as the neighbor discovery problem where all the n mobile nodes in the network may have different discovery schedules and relative phase offsets, and the maximum discovery latency is required to be less than or equal to L.

We now investigate the performance of U-Connect under asymmetric operation.

4.3.1 Analysis

Consider two nodes i and j with U-Connect discovery schedules $\psi_u^i(i,t)$ and $\psi_u^j(j,t)$. Let the relative phase offset between the nodes i and j be given by $\Phi_{i,j}$. Given a time instant t_0 with respect to the origin of time on node i, $\psi_u^i(i,t_0)$ denotes whether node i is in discovery mode, and $\psi_u^j(j,t_0+\Phi_{i,j})$ determines whether node j is in discovery mode.

Let p_i and p_j denote the distinct $primes$ used in the construction of $\psi_u^i(i,t)$ and $\psi_u^j(j,t)$. $\psi_u^i(i,t)=1$ when $[t]_{p_i}=0$, and $\psi_u^j(j,t)=1$ when $[t]_{p_j}=0$. If $p_i=p_j$, then the system is symmetric in nature and the asymmetric operation is not relevant.

For any given phasing $\Phi_{i,j}$, there is a single value of t every p_ip_j units of time, such that $[t]_{p_i}=[t+\Phi_{i,j}]_{p_j}=1$. This result follows from the Chinese Remainder Theorem [12] since p_i and p_j are distinct and pairwise co-prime by virtue of being primes themselves. Therefore, it follows that the worst-case discovery latency is given by p_ip_j. The worst-case discovery latency happens when the nodes encounter each other just after time t_1 such that $[t_1]_{p_i}=[t_1+\Phi_{i,j}]_{p_j}=1$, and the nodes are never simultaneously in discovery mode till time $t_1+p_ip_j$, when $[t_1]_{p_ip_j}=[t_1+\Phi_{i,j}]_{p_ip_j}=1$. It should be noted here that each node m is also in discovery mode at time t such that $0<[t_m]_{p_m^2}<\frac{p_m+1}{2}$. The average-case discovery latency for nodes i and j could of course be lower than p_ip_j.

The worst-case discovery latency of the Disco protocol for nodes i and j operating with prime pairs (p_{i_1},p_{i_2}) and (p_{j_1},p_{j_2}) is given by $p_{i_1}p_{j_1}$. However, the nodes may have picked the same value for their first prime i.e $p_{i_1}=p_{j_1}$, resulting in a worst-case latency of $\min(p_{i_1}p_{i_2},p_{i_1}p_{j_2})$. For more details please see [7].

Under the Quorum protocol, the worst-case discovery latency for nodes i and j operating with matrix sizes $m_i\times m_i$ and $m_j\times m_j$ is given by $\max(m_i^2,m_j^2)$. This is due to the fact that the node with the lower duty cycle determines the worst-case latency in Quorum.

Using the above analysis, when a node i with energy budget $O(\frac{1}{p})$ tries to discover another node j with much lower energy budget $O(\frac{1}{pr})$ (i.e. r times lower energy budget with $r>>1$), U-Connect and Disco will both result in a worst-case discovery latency of p^2r, whereas Quorum would be worse by a factor of r with a worst-case discovery latency of p^2r^2. U-Connect would therefore achieve at least the same performance as Disco under asymmetric operating conditions. However, in symmetric scenarios its asymptotic behavior is much better as shown earlier (refer section Sections 3.4 and 4.1), and under non-asymptotic conditions as shown in Figure 5. We envision U-Connect to be an unified protocol that can be applied to both symmetric and asymmetric scenarios.

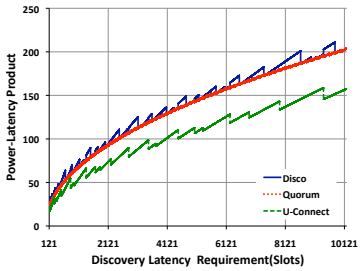


Figure 2: Power-Latency product of the different neighbor discovery protocols at various discovery latency requirements for *symmetrically* duty-cycled node pairs

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of U-Connect relative to earlier work. In particular, we focus on the energy consumption of different protocols under both symmetric and asymmetric operation for various discovery latency requirements. In order to avoid tying our results to particular hardware implementation platforms, this section uses (i) *percentage of system active time (duty cycle)* as the measure of energy consumption, and (ii) *number of slots* to describe discovery latencies. Actual power consumption values and slot durations are provided in later sections that describe our implementation.

5.1 Symmetric Operation

Applications comprising of homogeneous hardware platforms, such as the one described in this paper, are normally configured to be the same and are therefore concerned only with the symmetric mode of operation. All mobile sensor nodes in such systems have similar battery capabilities, and hence duty-cycle requirements. Given a latency requirement for neighbor discovery, we are interested in neighbor discovery protocols that can achieve low-energy operation. Figure 2 shows the power-latency product for Disco, Quorum, and U-Connect at various discovery latency requirements. We focused on latency requirements ranging from $121=11*11$ to $10201=101*101$, reflecting a duty-cycle ranging from 10% to 1% for a theoretical ideal protocol. For each data point, optimal balanced prime pairs were chosen for Disco, and optimal matrix sizes were chosen for Quorum. The results show that U-Connect clearly achieves lower-energy

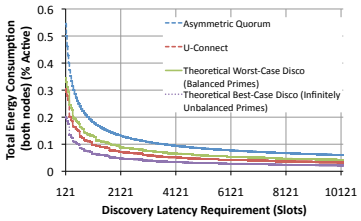


Figure 3: Normalized Energy Consumption (% Active) of the different neighbor discovery protocols at various discovery latency requirements for *asymmetrically* duty-cycled node pairs with a duty-cycle ratio (R) of 2

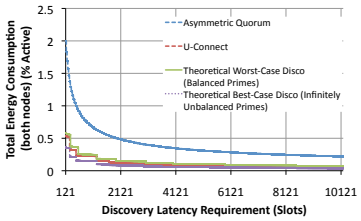


Figure 4: Normalized Energy Consumption (% Active) of the different neighbor discovery protocols at various discovery latency requirements for *asymmetrically* duty-cycled node pairs with a duty-cycle ratio (R) of 10

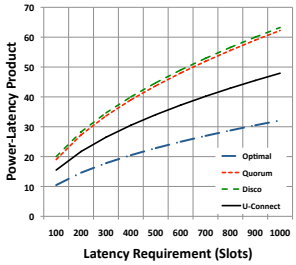


Figure 5: Performance comparison of U-Connect with other protocols under non-asymptotic conditions.

operation compared to Quorum and Disco. This result closely follows the analytical results provided in earlier sections with an asymptotic energy savings of 25%. U-Connect is therefore an excellent candidate for implementing neighbor discovery in systems with symmetric operating requirements.

5.2 Asymmetric Operation

Achieving asymmetric operation enables efficient usage of heterogeneous hardware platforms such as infrastructure nodes in the presence of mobile nodes. The asymmetric requirement is captured by an asymmetry ratio R, where

$$R = \frac{duty\ cycle\ of\ higher\ duty\ cycle\ node}{duty\ cycle\ of\ lower\ duty\ cycle\ node}$$

Figures 3 and 4 show the energy consumption of different neighbor discovery protocols at $R = 2$ and $R = 10$, for various discovery latency requirements. It is no surprise that the Quorum protocol consumes the most energy, since it was never originally intended to be deployed in asymmetric conditions. The performance comparison with the Disco protocol is trickier. Disco can be customized to either use balanced prime pairs or unbalanced prime pairs. Using extremely balanced prime pairs improves the symmetric discovery latency of Disco, while using extremely unbalanced prime pairs proves better for asymmetric discovery latencies. However, using unbalanced primes would result in poor performance under symmetric conditions. U-Connect has a single parameter p, which can be specified commonly across symmetric and asymmetric operations. As seen in Figure 3, U-Connect reduces the energy consumption below the theoretical worst-case of balanced prime pairs for Disco. However, infinitely skewed unbalanced primes provide a strict lower bound by achieving the theoretically optimal best-case for asymmetric operation. It should be noted that using such asymmetric pairs would cause the discovery latency to be extremely long when the other node also picks the same primes. As the ratio R is increased to 5 and subsequently to 10, the area between the best-case and worst-case for the Disco protocol decreases, and the performance closely follows that of U-Connect. In systems with strictly asymmetric operation, U-Connect may not reduce the energy consumption beyond Disco. However, in systems with symmetric operation, U-Connect achieves much better performance.

6. IMPLEMENTATION

We now describe our implementation and practical considerations of U-Connect. We have used U-Connect for a social networking application called *Friend Finder* as part of the Sensor Andrew [20] project at Carnegie Mellon University.

Sensor Andrew is a multi-disciplinary campus-wide scalable sensor network that is designed to host a wide range of sensing and low-power applications. The goals of Sensor Andrew are to support ubiquitous large-scale monitoring and control of infrastructure in a way that is extensible, easy to use, and provides security while maintaining privacy. Target applications currently being deployed include infrastructure monitoring, social networking, location tracking and building power monitoring. The *Friend Finder* application is described in Section 7.1.

6.1 FireFly Badge hardware platform

The social-networking application requires hardware that is light-weight, small and rechargeable. For this purpose, we designed a mobile sensor networking hardware platform called *FireFly Badge*. The name comes from the fact that these devices can be embedded into ID cards, badges or made into smaller "cards" that can be inserted into mobile phones. Figure 6 shows the hardware platform next to its enclosure and a U.S. quarter dollar coin to facilitate size comparison. Figure 7 shows the block diagram of the hardware. The *FireFly Badge* consists of an Atmega1281 [4] 8-bit processor running at 7.37MHz and a Chipcon CC2420 [22] 802.15.4 radio. It has a USB interface for charging and connecting to a PC or a laptop. The device can also be charged using a DC wall adapter. It also has an ADXL330 accelerometer that can

Figure 6: The FireFly Badge mobile sensor networking hardware platform with its enclosure and a quarter-dollar coin for size comparison.

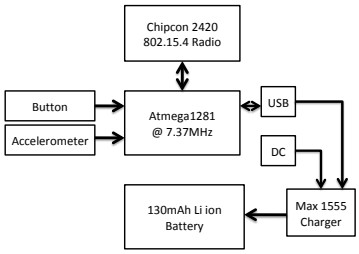


Figure 7: Block Diagram of FireFly Badge.

detect motion. A 3.7V Li-Ion battery powers the device.

6.2 U-Connect Operation and Implementation

The U-Connect schedule consists of two periodic operations. Let the prime number chosen by U-Connect based on the duty-cycle requirement p. According to the schedule defined in Equation 13, a node will be in *discovery mode* for a slot every p slots and for $\frac{p+1}{2}$ slots every p^2 (hyper-cycle) slots. In the implementation, the node listens at the slots that are multiples of p. This operation is fundamentally *Low Power Listening* (LPL) [19], where the listen period is p slots. The node transmits continuously for $\frac{p+1}{2}$ slots for every hyper-cycle. This periodic transmission is called *Low Power Transmit* (LPT). Clustering the transmission slots in this fashion, enables our implementation to spread the packet transmission over multiple slots, thereby enabling much lower slot sizes. When any two nodes i and j are considered, this implementation still achieves the same worst-case discovery latency of p^2, since either the transmission slots of node j will intersect with the receive slot of node i (when $[\Phi_{i,j}]_p < \frac{p+1}{2}$) or the transmission slots of node i will intersect with the receive slot of node j (when $[\Phi_{i,j}]_p \geq \frac{p+1}{2}$) (from cases 1 and 2 of Lemma 1). This energy-efficient implementation choice is consistent with the properties established in Section 4.

The protocol has been implemented on nano-RK [1] , a fully preemptive reservation-based [10] real-time operating system (RTOS) for sensor networks. The U-Connect protocol is implemented as a periodic task (τ) and assigned the highest priority. The period of the task τ is determined by the duty-cycle requirement. Although the minimum channel duration from the CC2420 datasheet [22] is 128 μs, we observed that

[1]www.nanork.org

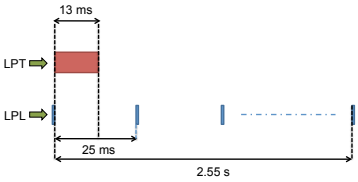


Figure 8: The U-Connect protocol operation for a prime number of 101 i.e. a duty-cycle of 1.5%. The slot size is 250μs.

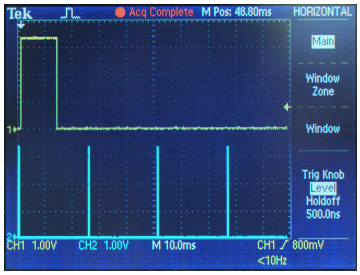


Figure 9: Protocol operation as seen on an oscilloscope connected to FireFly Badge hardware. The top line shows the LPT operation and the bottom line shows the LPL operation.

for a reliable channel detection, 250 μs is required. Hence, we have chosen a slot size of 250μs. At the beginning of every period of task τ i.e. p slots, the radio receiver is switched on and a CCA check for 250 μs is done. If the channel is busy, then receive the packet. For every p periods i.e. p^2 slots, a transmit of the node id is done. After a receive or transmit of a packet, a corresponding signal is sent to any tasks that have registered for the signal. The protocol operation is illustrated in Figure 8. Figure 9 shows the radio on time for LPT and LPL operation obtained from an oscilloscope connected to a FireFly Badge.

Disco uses a slot size of $10ms$ during which the radio is kept *ON*. During this slot, a beacon transmission is done at the beginning and end of the slot. This beaconing ensures that two nodes will discover each other regardless of how their slots overlap. U-Connect's slot size is much smaller (40 times) than Disco's slot size, achieving much higher energy savings and lower latencies.

6.2.1 *Interoperability*

In applications where the mobile nodes use infrastructure support, interoperability of the neighbor discovery protocol with the MAC protocol used by the infrastructure nodes for data flows is a desired feature. For example, the networks deployed as part of Sensor Andrew use BMAC as the MAC protocol. U-Connect achieves interoperability with LPL-based MAC protocols such as B-MAC by transmitting for duration equal to BMAC's check-period, whenever a packet needs to be sent to an infrastructure node. Also U-Connect will be able to receive any BMAC packet as long as the listen period of U-Connect, i.e. $p * 250\mu s$, is less than the check period used by BMAC.

6.2.2 Slot Non-Alignment

In practice, slots will rarely be aligned since nodes run independently and do not adjust for clock skews or set up a global time-reference. We need to ensure that two nodes will discover each other regardless of how their slots overlap. In U-Connect, when in LPT mode of operation, an unmodulated preamble is transmitted for p consecutive slots. The actual data, which is the node id in our implementation, is sent at the end of this preamble. Any other node that wakes up during this LPT will find the channel busy and will receive the data at the end of the preamble. When this happens, both nodes will attempt to transmit at the same time resulting in a collision. To avoid collisions in our protocol a CCA check is done before transmitting. Also, since the probability of two nodes being exactly phase-aligned to come in contact is low, we have not taken any explicit measures to overcome this.

6.2.3 Effects of clock drift

Clock drift is a major problem for many synchronizing protocols. These protocols need a mechanism for correcting the errors accumulated due to the clock drift. Some time-synchronization protocols use linear regression to perform this function [21, 16]. U-Connect operates in an unsynchronized manner and hence this does not affect the functioning of the protocol. The drift in clock could go beyond a slot duration and cause a phase shift. Our protocol compensates for these phase shifts and guarantees a discovery within a bounded time-period. The phase shifts could cause a delayed discovery though. In our platform, we have used an Abracon Corporation crystal [2] of 15ppm, which allows a phase shift of 1 slot duration ($250\mu s$ to occur after 17 seconds. In our implementation, we have set p equal to 101 giving a duty cycle of 1.5%. For this duty-cycle the worst case discovery latency (without packet loss) is 2.55 s, which is much less than 17 s.

6.3 Node Lifetime

In this section, we calculate the node lifetime α given the parameters in Table 1. The values are taken from the datasheet for our hardware platform, FireFly Badge.

Equation (18) represents the node lifetime for U-Connect. We do not consider the idle and sleep power-consumption of the radio as it is considerably less than the transmit and listen power-consumption of the radio in the equation. For a duty-cycle of 1.5% ($p = 101$), we obtain the node lifetime when using U-Connect from Equation 18 to be 387.44 hours which is approximately 15.5 days. In order to measure the current consumption of our hardware platform, we measure the voltage drop across a sense resistor of $10ohms$ connected in series between one of the power input pins and a regulated power supply providing a steady voltage of $3.3volts$. The measured current consumption is $26mA$ and $29.4mA$ during the transmission and receive slot durations. When the node is not transmitting or receiving, it is put to sleep immediately during which the current consumption is approximately $100\mu A$. Hence the actual node lifetime will be approximately 14 days. The increase in current consumption over the radio and active cpu consumption can be attributed to the other components in our hardware.

Parameter	Symbol	Value
CPU Active Current	I_{cpu}	7 mA
Radio Transmitter	I_{tx}	17.4 mA
Radio Receiver	I_{rx}	19.7 mA
Battery Capacity	B_c	130 mAh
Battery Voltage	B_v	3.7 V
Operating Voltage	V	3.3 V
U-Connect Prime	p	

Table 1: Parameters for determining node lifetime α

$$C = B_c \times B_v \tag{17}$$

$$\alpha = \frac{C}{V \times \{\frac{I_{tx}}{2p} + \frac{I_{rx}}{p} + \frac{3I_{cpu}}{2p}\}} \tag{18}$$

7. EXPERIMENTAL RESULTS

In this section, we experimentally evaluate the performance of U-Connect based on our implementation. In the following experiments, 20 nodes were first programmed to operate at a duty-cycle of approximately 1.5%, and then for a duty-cycle of 5% i.e. the U-Connect prime is 101 and 31 respectively. As described earlier, the slot duration was maintained at 250μs. The neighbor discoveries were made by the node connected to the PC by a USB port, where experimental results were logged. This node is called 'Control Node'. For obtaining two-node discovery latencies, every time a discovery of a particular node is done, a kill packet is sent to all the nodes by the Control Node. Each data point is obtained over the measurements from 100 individual experiments.

In our first set of experiments, all nodes were allowed to synchronize, whenever they discovered each other. Figure 10 and Figure 11 shows the discovery latency between the 'Control Node' and another *uniquely designated node*, when other nodes are added within the same collision-domain operating with primes of 101 and 31 respectively. When no other nodes are present in the collision domain, the discovery latency is quite close to the theoretical scenario of approximately $101*101 = 10201$ slots and $31*31 = 961$ slots. As the number of nodes within the same collision domain increases until a certain point, the *expected time* for the *designated pair* of nodes to synchronize decreases. When the number of nodes increase beyond 9 and 6 for primes of 101 and 31 respectively, the probability of collision increases thereby increasing the discovery latency of the nodes. The average variance of the discovery latency under primes of 101 and 31 is 808 and 127 respectively.

In our next set of experiments, we evaluated the time taken by the node to discover all its neighbors. Figure 12 shows the discovery latencies for the 'Control node' to discover all

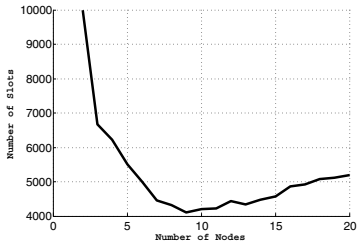


Figure 10: This plot shows the discovery latencies for a pair of nodes as the number of nodes in their vicinity is increased up to 20 nodes. The nodes are operating with a u-connect prime of 101

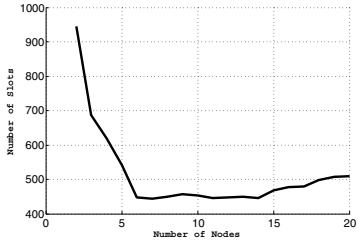


Figure 11: This plot shows the discovery latencies for a pair of nodes as the number of nodes in their vicinity is increased up to 20 nodes. The nodes are operating with a u-connect prime of 31**.**

the nodes within its vicinity, as the number of neighbors increases. We restricted our experimental results to interesting regions with a non-trivial number of nodes. Observe that there are two counteracting effects in this evaluation. Adding more neighboring nodes increases the chance for early synchronization, thus acting in the favor of early discovery. However, adding more neighboring nodes increases the total number of nodes to be discovered thereby adversely increasing the discovery latency. It can be seen that the discovery latencies quickly drop off as more nodes are added to the same collision domain until about 9 nodes. This shows that as the number of nodes increase, the synchronizing effect dominates the cumulative discovery latency. As the number of nodes is increased beyond 9, we see that the discovery latency is starting to increase slowly. The discovery latency is increasing because adding more nodes to the same collision domain increases the probability of collisions, therefore, the discovery latency starts increasing and if more nodes are added, the discovery latency might go beyond the theoretical limit. In the following section, we will describe an application that uses the neighbor discovery service provided by U-Connect.

7.1 Friend Finder

Neighbor discovery is used in a wide variety of applications, which involve mobile-to-mobile and/or mobile-to-static node communication. In this section, we demonstrate a *friend-finder* application that uses U-Connect. This application was developed as part of *Sensor Andrew*. Figure 13 shows the

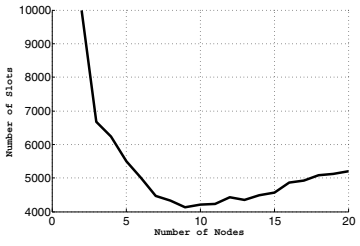


Figure 12: This plot shows the discovery latencies for a node to discover all the nodes in its vicinity as the number of nodes is increased up to 20 nodes. The u-connect prime is 101**.**

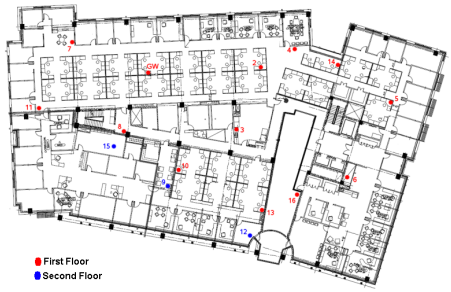


Figure 13: This figure depicts the sensor network deployment in CIC building at Carnegie Mellon university.

deployment at the Collaborative Innovation Center (C.I.C) in Carnegie Mellon University. We now describe three key capabilities of *friend-finder* that use the neighbor discovery service provided by U-Connect.

(1) Friend Alert: The idea behind friend alert is to enable networking opportunities for students in the physical world. Students are provided with the *FireFly Badge* in a key-chain form factor. Friends list with 8-byte unique ids can be downloaded to the badge. Whenever the U-Connect neighbor discovery service detects a neighboring sensor node, it notifies the Friend Finder application. Based on the unique id, if the neighboring sensor node is present in the list of friends, then the *FireFly Badge* issues a visual notification through the LED interface. U-Connect thus enables the *Friend Alert* capability, which serves a means for physical networking. We are also investigating better user interfaces and privacy issues in this feature.

(2) Suggest Friends: Complementing the *Friend Alert* feature results in the capability to **Suggest Friends**. The U-Connect neighbor discovery service logs the unique ids of nodes in its vicinity. Based on interaction durations, the most persistent set of node ids is cached in the *FireFly Badge*. When the user discovers any infrastructure node, this cached set of ids gets published to the corresponding user's interaction logs. Based on the interaction logs, friends for online social networking are suggested from physical networking information. It is key to note that the interoperability of U-Connect with B-MAC enables the discovery of infrastructure nodes that operate at a much lower duty cycle. In our

scenario, the mobile nodes are rechargeable and can therefore operate at a higher duty cycle (p is 101 slots or $25ms$). On the other hand, the infrastructure nodes are deployed for months together, and therefore require a lower duty cycle (BMAC check rate of $100ms$). Upon detecting an infrastructure node packet (owing to higher check rate of $25ms$), the mobile badge decodes the BMAC check rate from the packet header, and responds with a longer duration LPT ($100ms$) followed by id information.

(3) Find My Friend: Another useful capability of *Friend Finder* is the *Find My Friend* option. It provides coarse-grained localization using the *last interacted infrastructure node* information. Whenever a mobile node is discovered through U-Connect, the infrastructure node timestamps the interaction and logs it to the corresponding user information. Authorized friends can query this information and determine the last known location of the user through the most recent timestamp.

8. CONCLUSION

In this paper, we have provided a theoretical formulation of the asynchronous neighbor discovery problem, and developed the power-latency product as a metric for evaluation of neighbor discovery protocols for mobile sensor networks. We have proposed a new neighbor discovery protocol called U-Connect. We have analytically shown that U-Connect is an 1.5-approximation algorithm for the symmetric asynchronous discovery scenario, and that existing protocols such as Quorum and Disco are 2-approximation algorithms.

The U-Connect implementation uses what we call *Low Power Transmit* along with Low Power Listening, the periods consisting of a *prime* number of slots, that achieves asynchronous symmetric and asymmetric discoveries at minimal and predictable energy costs and latency. We have also evaluated the performance of U-Connect against existing protocols through simulations and experiments. We implemented the U-Connect protocol on our custom-designed mobile sensor networking platform called FireFly Badge. Our implementation uses a slot duration of only 250 μs achieving orders of magnitude lower latency for a given duty cycle compared to existing neighbor discovery schemes for wireless sensor networks.

The U-Connect protocol enables multiple applications ranging from detecting social interactions to location tracking. It is a unified protocol in the sense that it performs well under both the symmetric and asymmetric scenarios. We have demonstrated the capabilities of U-Connect in the context of the *Friend-Finder* application that has been deployed as part of Sensor Andrew, a campus-wide sensor network.

9. ACKNOWLEDGMENT

We would like to thank Dr. Anthony Rowe for providing technical guidance on the hardware design of FireFly Badge. We would also like to thank Dr. Luca Mottola for shepherding our paper and providing useful comments to improve this paper.

10. REFERENCES

[1] www.bluetooth.com.

[2] Abracon Corporation. *ABMM Crystal 7.37MHz*, 2008.

[3] I. Anderson. *Combinatorial Designs and Tournaments*, chapter 2. Oxford University Press, 1998.

[4] Atmel Corporation. *Atmega 1281 Datasheet*, 2008.

[5] C.-M. Chao, J.-P. Sheu, and I.-C. Chou. An adaptive quorum-based energy conserving protocol for ieee 802.11 ad hoc networks. *Mobile Computing, IEEE Transactions on*, 5(5):560–570, May 2006.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001.

[7] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84. ACM, 2008.

[8] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 87–101. ACM, 2007.

[9] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. volume 5, pages 3418–3423 vol.5, 2002.

[10] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: an energy-aware resource-centric rtos for sensor networks. In *26th IEEE International Real-Time Systems Symposium*, pages 10 pp.–265, Dec. 2005.

[11] J.-H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.

[12] H. L. Ivan Niven, Herbert S. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley and Sons, 1991.

[13] A. Kandhalu, K. Lakshmanan, and R. Rajkumar. Neighbor discovery in mobile sensor networks. Technical report, Carnegie Mellon University, 2010.

[14] S. Lai. *Heterogenous Quorum-based Wakeup Scheduling for Duty-Cycled Wireless Sensor Networks*. PhD thesis, Virginia Polytechnic Institute and State University, 2009.

[15] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 256–269. ACM, 2004.

[16] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49. ACM, 2004.

[17] M. J. McGlynn and S. A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145. ACM, 2001.

[18] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[19] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[20] A. Rowe, M. Berges, G. Bhatia, E. Goldman, R. Rajkumar, L. Soibelman, J. Garrett, and J. M. F. Moura. *Sensor Andrew: Large-Scale Campus-Wide Sensing and Actuation*. Carnegie Mellon University, 2008.

[21] H.-S. W. So, G. Nguyen, and J. Walrand. Practical synchronization techniques for multi-channel mac. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 134–145. ACM, 2006.

[22] Texas Instruments. *Chipcon CC2420 Datasheet*, 2008.

[23] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. volume 1, pages 200–209 vol.1, 2002.

[24] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.

[25] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 35–45. ACM, 2003.