RBTP: Low-Power Mobile Discovery Protocol through Recursive Binary Time Partitioning

Dong Li, *Student Member*, *IEEE*, and Prasun Sinha, *Senior Member*, *IEEE*

Abstract—With increasing prevalence of mobile wireless devices with WiFi and Bluetooth capability, new applications that can make use of limited contact opportunities when the devices are physically close are emerging. Proximity-based social networking, and location specific dissemination of advertisements and events, are some such applications. Discovering such services is a challenging problem due to energy budget limitations, user mobility, and nonuniformity and the time-varying nature of energy budgets across users. It is important to rapidly discover such mobile services to make use of limited contact opportunities. To support such applications, we seek to design a localized discovery scheme that can minimize the expected contact latency between mobile phones with limited energy budgets. All the existing neighbor discovery schemes assume lack of any time synchronization. However, in practice sufficiently accurate time synchronization can be achieved with existing time synchronization techniques. We propose *Recursive Binary Time Partitioning* (RBTP), a scheme that determines how the devices should wake up and sleep to achieve minimal contact latency with other nearby devices. RBTP achieves provable performance bound and outperforms state-of-the-art asynchronous protocols for smartphones. When compared with the optimum scheme, the contact latency is shown to be within a factor of $9/8$ in the expected case and 2 in the worst case.

Index Terms—Neighbor discovery, mobile devices, energy management in smartphones

1 Introduction

Smartphones with local-area networking interfaces, such as WiFi and Bluetooth, are becoming increasingly popular. These networking interfaces make the phones capable of mutual discovery and information exchange when they are within communication range of each other. For example, people can receive electronic coupons as long as they are present in or pass by certain stores [33]; LinkedIn [15] can suggest people to link each other if they meet at a conference; and, Facebook [9] can recommend people who meet frequently with each other on weekends as friends. Such services have the potential to bridge the virtual cyberspace with our physical world.

Implementing such applications using periodic GPS updates has its own shortcomings. The first reason is that GPS is power hungry. Paek et al. [22] and Ben Abdesslem et al. [4] both found that the battery life is between 7 and 11 hours when the GPS is always on. The second reason is that GPS may not be available in indoor environments. In addition, people are concerned about their location privacy when reporting their GPS coordinates to servers.

These applications will typically run in the background waiting to discover neighboring phones, access points (APs), or other services without the need to interact with users for extended periods, especially when the density of devices is low. If WiFi is being used for such purposes without proper sleep scheduling of the device interface, the phone's battery will quickly drain out. We conducted an experiment using an HTC G1 phone [10] with a fully charged 1,150-mAh battery with the WiFi interface enabled and the cellular interface turned off. The phone sends a small packet (containing ID and time stamp, 14-byte payload) per second in the 802.11 ad hoc mode. We observed that the battery lasted for only 5.5 hours. In contrast, the standby time is longer than 48 hours when WiFi was turned off and the cellular interface was turned on. Today, typical applications and usage patterns of smartphones do not require using the WiFi interface that often. However, to support applications that can exploit the results of neighbor discovery, the phones will need to probe their surroundings at a certain minimum rate, which necessitates the design of improved techniques for managing energy consumption of the WiFi interface. Thus, smarter mechanisms for battery usage are needed.

Mobile service discovery on smartphones is challenging due to multiple reasons. First, users may like to limit the energy expended by the neighbor discovery service because it can affect the battery availability for other primary tasks of the phone, such as making or receiving a phone call. Second, the energy budget itself may vary between users, and for each user it may vary with time due to other uses of the device. This makes it challenging to simultaneously optimize the operation of all nodes, especially because the pairs of nodes that will come in contact in the future is not known a priori. And, third, as user mobility is difficult to model and predict, the contact patterns, i.e., the time and duration of contacts, are not known beforehand.

The goal of our research is to design a localized discovery scheme that minimizes the expected contact latency between smartphones with limited energy budgets. The *contact latency* is the duration from the time a smartphone comes into

- *The authors are with the Department of Computer Science and Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210-1277. E-mail: {lido, prasun}@cse.ohio-state.edu.*

Manuscript received 11 Oct. 2011; revised 4 June 2012; accepted 3 Nov. 2012; published online 20 Nov. 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-10-0559. Digital Object Identifier no. 10.1109/TMC.2012.240.

another phone's contact range, to the time the two phones mutually discover each other. The mutual contact range is defined as the range in which both phones can communicate with each other. Optimizing the contact latency also optimizes the *missing rate*, which is the probability that two devices that are in each other's contact range fail to discover each other during the period in which they were in their mutual range. If the contact latency is high, the *missing rate* is expected to be high as well.

Previous works [16], [6], [13], [3] on neighbor discovery of mobile devices assume that the clocks of the devices are not synchronized. They adopt a slotted time model in which time is divided into equal-sized slots. A device is scheduled to wake up or sleep in a slot deterministically [6], [13], [3] or probabilistically [16], [3]. Devices in their mutual contact range are assumed to be able to discover each other when two wake-up slots overlap. These protocols are mainly designed for sensor networks where time synchronization is challenging but the devices can quickly switch the network interface to different states. We show that the protocols have poor performance in terms of both high contact latency and high missing rate in our simulations (Sections 6.2.1 and 6.2.2). In contrast, a smartphone has the luxury of synchronizing its clock through GPS, Internet, or cellular protocols. *Partial synchronization or synchronization with limited inaccuracy can be exploited to design better neighbor discovery protocols.*

The proposed scheme, called Recursive Binary Time Partitioning (RBTP), is a synchronized protocol comprised of the number and the patterns of wake ups for the phones. Within a time period, the number of wake-up instances depends on the phone's own energy budget. RBTP wakes up the network interface at certain times and puts it to sleep until the next wake-up instance. In a typical smartphone, the WiFi chipsets already have implemented the functions to quickly switch between wake-up and sleep modes. For example, in the 802.11 protocol power saving mode, the mobile device can choose its sleep period in multiples of beacon intervals, and can wake up at the beginning of a beacon interval for a fixed duration of time, which is called the Announcement Traffic Indication Messages (ATIM) window. The ATIM window is smaller than a beacon interval Gast [11]. Because a typical beacon interval is 100 milliseconds, the WiFi module should be able to switch between wake-up and sleep modes in 100 milliseconds. However, these functions are provided by proprietary binary drivers or firmwares and are unavailable for user-space applications. Typically, the user-space APIs for switching the WiFi interface take longer time. For example, Bakht et al. [3] found that the time to switch the wireless interface is around 1 second for Nokia N900. In contrast, implementations on sensors such as by Dutta and Culler [6] and Kandhalu et al. [13] use 10 milliseconds and 250 microseconds as their slot sizes, respectively.

Smartphones often have inaccurate clocks due to the low quality of the clocks and infrequent synchronization events. For example, we found that the clock difference between two smartphones increases by as much as 5 seconds every 2 hours. Some modules in a smartphone such as GPS [12] can provide accurate clock. GPS is power hungry [4], [22]. Moreover, in off-the-shelf smartphones, these modules are wrapped by proprietary binary drivers or firmwares, including in the open-source Android phones. The drivers and firmwares are provided by hardware manufactures and are often closed source. For this reason, a smartphone's operating system clock is independent of these modules and generally not accurate without calibration. We implemented two clock synchronization methods based on Network Time Protocol (NTP). In our experiments, a smartphone only needs to connect to a public NTP server every 6 hours to calibrate its clock and keep the clock error within 100 milliseconds. The energy overhead for this operation is negligible.

This paper makes the following contributions:

- We propose *Recursive Binary Time Partitioning*, a scheme that can achieve near optimum performance and that can be practically realized with local information. Our design allows the devices to adapt their number of wake-up instances *independently* based on their respective energy limitations.
- We prove that the contact latency of RBTP is bounded within a factor of $9/8$ and 2 from the optimum solution in the average case and the worst case, respectively.

The rest of the paper is organized as follows: Section 2 presents the model. Section 3 discusses the optimum solution when the phones have perfectly synchronized clock. Section 4 presents the RBTP scheme. Section 5 shows implementation of our clock synchronization mechanism in smartphones. Section 6 compares RBTP with recently proposed neighbor discovery protocols using simulations. Section 7 introduces previous works related to this research. Section 8 discusses the challenges related to designing neighbor discovery applications and the future work. Conclusion is presented in Section 9.

2 Model and Problem Statement

2.1 Model Description

In the rest of the paper, we will use the term *node* in place of user, mobile device or a smartphone. The set of nodes is denoted by U. The *contact range* of a pair of nodes is the distance within which the two nodes can receive each other's beacons. Note that this definition of contact range does not require all nodes to use the same contact range.

Time is partitioned into frames of length T, which contains multiples of slots. The slot size is δ. We assume that the time difference between clocks on any two devices is bounded by the slot size δ. The number of slots in a frame is $N = \lfloor T/\delta \rfloor$. In each wake-up instance, a node wakes up for one slot of duration δ. It sends a beacon at the beginning and at the end of the slot so that two nodes can discover each other when their slots are overlapping but are not necessarily perfectly aligned. We assume that the beacons are reliable for the ease of description of our solution, but consider the impact of beacon losses in our evaluation in Section 6.2. The model of discovery is similar to the slotted time model considered in other works on neighbor discovery [6], [13]. T and δ are fixed global parameters across all the nodes. A node's duty cycle d is defined as the

fraction of time that a phone remains awake within a frame. We use the same definition of duty cycle as in previous works [6], [13], [3]. If a node u wakes up n times in a frame, its duty cycle is given by

$$d = n\delta/T. \quad (1)$$

2.2 Problem Statement

Assume u_i is an arbitrary node in U with a budget of n_i wake-up instances in each time frame. u_i has no knowledge of the number of wake-up slots of the other nodes. The key question is—*What is the optimal strategy that can minimize the average contact latency between any two nodes?*

3 The Optimum Solution

Within a frame, if two nodes u_i and u_j have n_i and n_j wake-up instances, respectively, then they have at most $n = \min(n_i, n_j)$ wake-up instances to meet. We obtain the optimum pattern as stated below in Theorem 3.1. Observe that a similar result was shown in [30], although their objective of minimizing the missing rate is slightly different from our objective of minimizing the contact latency. Later, we will show that this optimal scheme requires knowledge of the energy budget of the other nodes, and hence is not practically useful.

Theorem 3.1. *If n is known to two nodes with synchronized clock, and if the time they arrive into their mutual contact range is uniformly distributed within a frame, then uniform separation of the wake-up instances minimizes the contact latency.*

Proof. We consider a single time-frame that begins at time t_0 and ends at time $t_0 + T$. Two nodes u_i and u_j will arrive into their mutual contact range in this time frame. We first investigate an arbitrary scheme that allows u_i and u_j to have $n = \min(n_i, n_j)$ common wake-up instances, which is the maximum number of wake-up instances they could have in common. We assume these wake-up instances begin at time $t_1, t_2, \ldots, t_n$ and they are well separated so that the previous wake up can finish before the next wake-up instance begins. Without loss of generality, we set $t_n = t_0 + T$. The durations between these instances are $I_1 = t_1 - t_0$, $I_2 = t_2 - t_1$, $I_3 = t_3 - t_2, \ldots, I_n = t_n - t_{n-1}$. The probability that u_i and u_j arrive into the mutual contact range in the interval $[t_{k-1}, t_k]$ is proportional to I_k, where $k \in \{1, 2, \ldots, n\}$. This probability is given by I_k/T. If u_i and u_j arrive into their mutual contact range in interval $[t_{k-1}, t_k]$, the expected contact latency is $I_k/2$. So, the expected contact latency $\mathbb{D}_{exp}$ within this frame is

$$\mathbb{D}_{exp} = \sum_{k=1}^{n} \frac{I_k}{T} \times \frac{I_k}{2}. \quad (2)$$

Using $\sum_{k=1}^{n} I_k = T$ and the Cauchy-Schwarz inequality, we can get $\mathbb{D}_{exp} \geq T/2n$. When $I_1 = I_2 = \cdots = I_n$, $\mathbb{D}_{exp} = T/2n$, which is the optimal expected contact latency represented as follows:

$$\mathbb{D}_{exp}^{\mathrm{OPT}} = \frac{T}{2n}. \quad (3)$$

□

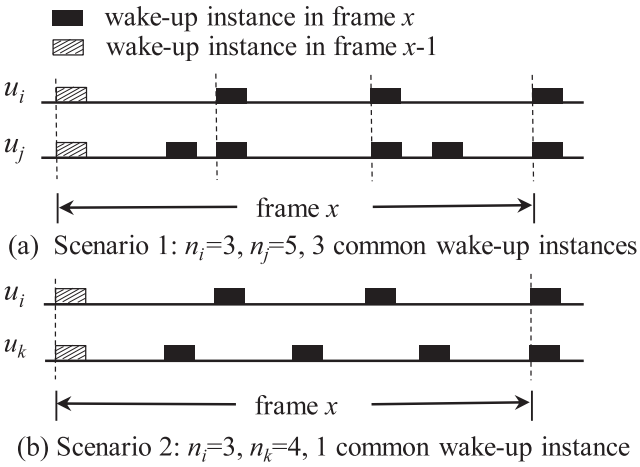


Fig. 1. (a) both u_i and u_j know n_i and n_j and their wake-up patterns a priori. (b) u_i and u_k have no a priori knowledge of each other's number or pattern of wake-up instances. The dashed vertical lines indicate the common wake-up instances of the two nodes.

In the worst case, two nodes will move to their mutual contact range at the beginning of the largest interval and then they have to wait until the end of that interval to discover each other. We denote the worst-case contact latency as $\mathbb{D}_{worst}$:

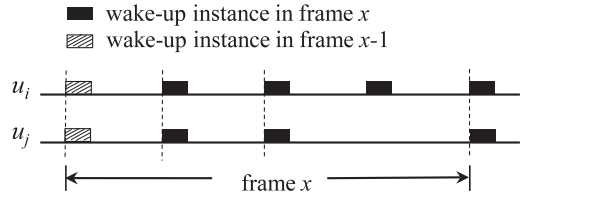
$$\mathbb{D}_{worst} = \max_{k=1,\ldots,n} \{ I_k \}. \quad (4)$$

Observe that the pattern described in Theorem 3.1 also minimizes $\mathbb{D}_{worst}$. The worst-case contact latency in the optimal scheme is

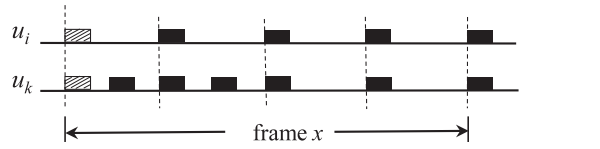
$$\mathbb{D}_{worst}^{\mathrm{OPT}} = \frac{T}{n}. \quad (5)$$

According to the model in Section 2.1, $T/n = \delta/d$. The above analysis indicates that the frame size does not affect $\mathbb{D}_{exp}^{\mathrm{OPT}}$ and $\mathbb{D}_{worst}^{\mathrm{OPT}}$ if the duty cycle d and the slot size δ are fixed. In the remaining paper, we fix the frame size to be $N = 1{,}024$ slots, thus $T = 1{,}024\delta$. Fig. 1a shows an example, where the minimum number of wake-up instances between the two nodes is 3. If two nodes know that they are going to meet in the future, but do not know the time when they will meet, then they can determine a pattern of wake up that minimizes the expected contact latency. Such a pattern is based on Theorem 3.1 and is shown in Fig. 1a. However, this is impractical to implement as it requires prior knowledge of which node pairs are going to meet next and their energy budgets. Without such knowledge, the scheme will not remain optimum. For example, in Fig. 1b when u_i comes in range of another node u_k with the energy budget of four wake-up instances, but with a different wake-up pattern than u_j, the optimality of contact latency is lost.

From this example, we observe that a good scheduling strategy should try to make two nodes meet for the maximum possible number of instances, i.e., $\min(n_i, n_j)$, where the wake-up instances are well separated within the frame. In the next section, we will present a localized scheme that achieves near-optimum performance both in terms of the expected and the worst-case contact latencies.



(a) Scenario 1: n_i=4, n_j=3, 3 common wake-up instances in frame x



(b) Scenario 2: n_i=4, n_k=6, 4 common wake-up instances in frame x

Fig. 2. RBTP for two scenarios. The dashed vertical lines indicate common wake-up instances of the two nodes.

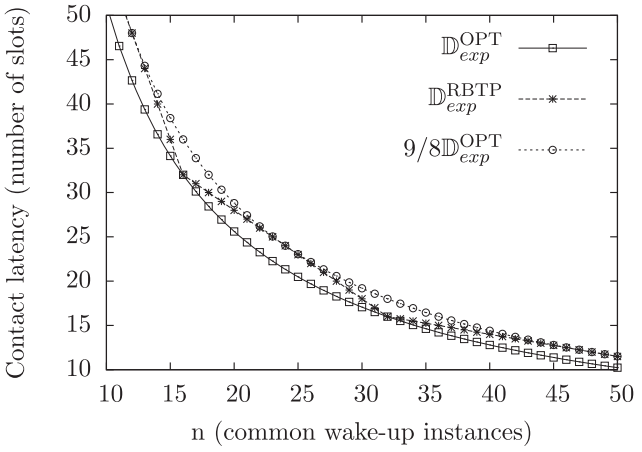


Fig. 3. The upper and lower bounds of the expected contact latency of RBTP between nodes u_i and u_j. $n = \min(n_i, n_j)$.

4 Recursive Binary Time Partition

The main idea of RBTP is that a node wakes up at instances that recursively partition a frame in a binary fashion. If a node has $n = 1$ wake-up instance in a frame, it wakes up at the end of the frame. When a node has $n > 1$ wake-up instances, it determines its wake-up time by calculating x and m using

$$n = 2^x + m,\ 0 \le m < 2^x, x \in \mathbb{N}, m \in \mathbb{N}. \tag{6}$$

Then, it partitions the frame into 2^x equal-length intervals with 2^x wake-up instances. RBTP places the remaining m wake-up instances in the middle of the first m intervals for the sake of simplicity. With this schedule, denote t_k as the start time of the kth $(k > 0)$ wake-up instance in the frame. When $n > 1$, t_k is calculated by (7), where t_0 is the starting time of the frame and T is the length of a frame. The node will switch to sleep mode at time $t_k + \delta$:

$$t_k = \begin{cases} t_0 + kT/2^{x+1}, & k \in \{1, 2, \ldots, 2m\} \\ t_0 + (k-m)T/2^x, & k \in \{2m+1,\ 2m+2,\ \ldots, n\}. \end{cases} \tag{7}$$

Observe that this scheme is completely local, as each node can determine its wake-up instances without knowledge of the energy budget of other nodes. In a frame, if u_i and u_j have n_i and n_j wake-up instances, respectively, they will have $n = \min(n_i, n_j)$ common wake-up instances in this frame. These n common wake-up instances overlap with the wake-up instances of the node with fewer wake-up instances.

Figs. 2a and 2b show two examples in which two nodes use RBTP to schedule their own wake-up instances. The two nodes in Fig. 2 can have three common wake-up instances, irrespective of the fact that they each have different number of wake-up instances.

4.1 Expected Contact Latency and the Worst-Case Contact Latency of RBTP

Using RBTP, two nodes u_i and u_j will have n common wake-up instances in a frame, where $n = \min(n_i, n_j)$. The expected latency can be calculated using the same mechanism as in the proof of Theorem 3.1. The length of the first $2m$ intervals is $T/2^{x+1}$, and the length of the last $n - 2m$ intervals is $T/2^x$. The expected contact latency of RBTP is as follows:

$$\mathbb{D}_{exp}^{\mathrm{RBTP}} = 2m\frac{(T/2^{x+1})^2}{2\,T} + (n-2m)\frac{(T/2^x)^2}{2\,T}. \tag{8}$$

Using (6) we can simplify (8) and get

$$\mathbb{D}_{exp}^{\mathrm{RBTP}} = \frac{T(2n-3m)}{4(n-m)^2} < \frac{9T}{16n}. \tag{9}$$

Inequality (10) shows the lower bound and the upper bound compared with $\mathbb{D}_{exp}^{\mathrm{OPT}}$:

$$\mathbb{D}_{exp}^{\mathrm{OPT}} \le \mathbb{D}_{exp}^{\mathrm{RBTP}} < \frac{9}{8}\,\mathbb{D}_{exp}^{\mathrm{OPT}}. \tag{10}$$

The expected latency of RBTP is the same as the optimal latency when n is a power of 2. Note that our scheme is local and within a factor of $9/8$ (or 12.5 percent) of the optimum. Note that the scheme considered to compute this ratio is an upper bound on the performance of the real optimum and it cannot be implemented in practice for all possible scenarios as argued earlier. Fig. 3 shows the expected contact latency of RBTP, $\mathbb{D}_{exp}^{\mathrm{OPT}}$ (lower bound), and $9/8\ \mathbb{D}_{exp}^{\mathrm{OPT}}$ (upper bound).

In the worst case, the contact latency of RBTP is $T/2^x$. From (6), we can derive $2^x > n/2$. So

$$T/2^x < 2(T/n). \tag{11}$$

By combing Inequality (11) and (5), we can find the upper bound of the worst-case contact latency of RBTP as shown below:

$$\mathbb{D}_{worst}^{\mathrm{RBTP}} < 2\ \mathbb{D}_{worst}^{\mathrm{OPT}}. \tag{12}$$

Inequality (12) indicates that the worst-case contact latency of RBTP is within a factor of 2 of the minimized worst-case contact latency.

Fig. 3 shows the expected contact latency of RBTP, the upper bound and the lower bound. Fig. 4 shows the worst-case contact latency of RBTP, the optimal worst-case contact latency (lower bound) and two times of the optimal worst-case contact latency (upper bound). In both the figures, the contact latency is measured in units of slots. Note that when

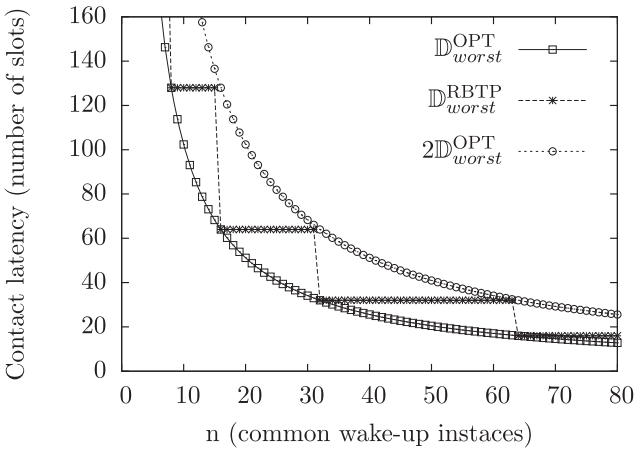


Fig. 4. The upper and lower bound of the worst-case contact latency of RBTP between nodes u_i and u_j. $n = \min(n_i, n_j)$.

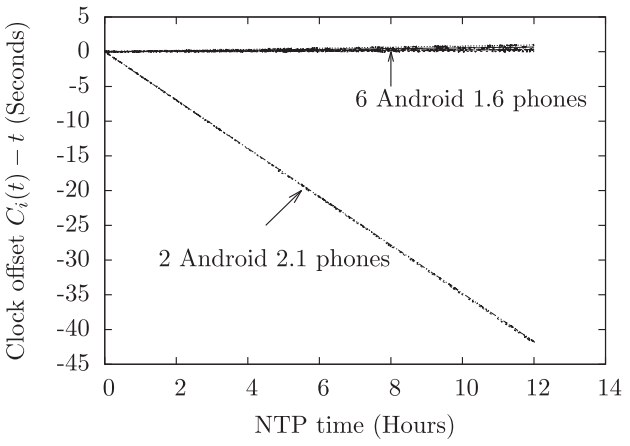


Fig. 5. Offset of the smartphone's clock as compared to the time provided by NTP.

the clock inaccuracy is higher than the slot size δ, the RBTP scheme does not preserve the near optimal performance.

Several asynchronous neighbor discovery protocols have a closed-form worst-case contact latency when all nodes use a fixed duty cycle. Bakht et al. [3] showed that when the duty cycle is $1/x$, the worst-cast contact latencies of Disco, U-Connect, and Searchlight are $4x^2$, $9x^2/4$, and x^2 slots, respectively. According to (1) and Inequality (11), the worst-case bound for RBTP is $2x$ slots, which is one order less than state-of-the-art deterministic asynchronous protocols. RBTP achieves low worst-case contact latency bound by exploiting the synchronized clock and the specially designed pattern.

5 Smartphone Clock Synchronization Experiments

In Section 5.1, we will show how clock synchronization in smartphones can be achieved using the NTP protocol. In Section 5.2, we will measure and estimate the energy overhead of synchronizing a smartphone.

5.1 Smartphone Clock Synchronization

In this section, we show that clock synchronization can be achieved with small energy overhead when using NTP. In theory, the clock can be synchronized using GSM, 3G or 4G protocols. However, these protocols are provided in proprietary binary modules and do not have an interface for application developers to synchronize the smartphone's clock. In contrast, NTP is more convenient to implement on the Android phones. We have implemented the linear fitting method and the linear programming method [18] based on NTP to calibrate the clocks.

The terminology used here to describe a clock is similar to [17]. A clock on a node i is represented by C_i. A "true" clock is a clock that always returns the "true time." Clock C_i returns time $C_i(t)$ at the true time t. The *offset* of clock C_i is defined as $C_i(t) - t$, i.e., the difference between clock C_i and the true time t. The *frequency* of clock C_i is defined as $C_i'(t)$, i.e., the rate at which the clock increases. The *skew* of clock C_i is defined as the difference between $C_i'(t)$ and the frequency of the true clock, which is 1 sec/sec.

We first conduct an experiment to show the skew of eight HTC G1 smartphones. Six of the phones have Android 1.6 OS and the other two have Android 2.1 OS. A set of public NTP servers were used in the experiment. The addresses of these NTP servers range from 0.north-america.pool.ntp.org to 3.north-america.pool.ntp.org. The mean of the round trip delays between the phones and the NTP servers is 107 milliseconds. In this experiment, each phone retrieves an NTP time stamp from one of the NTP servers every 30 seconds, and logs its local time and the retrieved NTP time. The offset of each phone against the NTP time is plotted in Fig. 5. The slope of the lines in Fig. 5 represents the skew of the clocks. From the figure, we can see that the offset linearly increases with time and the skews of the clocks do not vary significantly with time. In addition, Fig. 5 shows that the clocks of the phones with the Android 2.1 OS are slower than the clocks of phones with the Android 1.6 OS.

In our clock synchronization experiment, the smartphones are programmed to get m time stamps from the NTP servers in a synchronization instance. The synchronization instances are equally separated by H hours. Because the round trip delays between the smartphones and the NTP servers vary with time, multiple time stamps are retrieved in each synchronization instance. A phone i logs its local time $C_i(t)$ and the NTP time $C_i^{ntp}(t)$. The linear fitting and the linear programming methods are used to estimate the skew of a phone based on the logs. Let the vector of the local clock log of a phone be $\mathbf{C_i}$ and the corresponding NTP clock vector be $\mathbf{C}_i^{ntp}$. By fitting the linear function $\mathbf{C}_i^{ntp} = \text{skew} \cdot \mathbf{C}_i + b$, we can obtain the estimated skew. Moon et al. [18] provide another method that can find the skew by using linear programming. In our implementation, in each synchronization instance 10 NTP time stamps are retrieved and both $\mathbf{C}_i^{ntp}$ and C_i contain the time stamps from the previous two NTP synchronization instances.

To test the accuracy of the calibrated clocks, we send one packet from each smartphone to a laptop at the "same" time, and examine the time instance when the laptop receives the packets. The intuition is that if the smartphones are well synchronized, the reception time of the packets should be very close on the laptop. We assume that the packet delay from the smartphones to the laptop does not vary significantly with time. We set up an 802.11 ad hoc network with a Thinkpad T52 laptop and let the phones

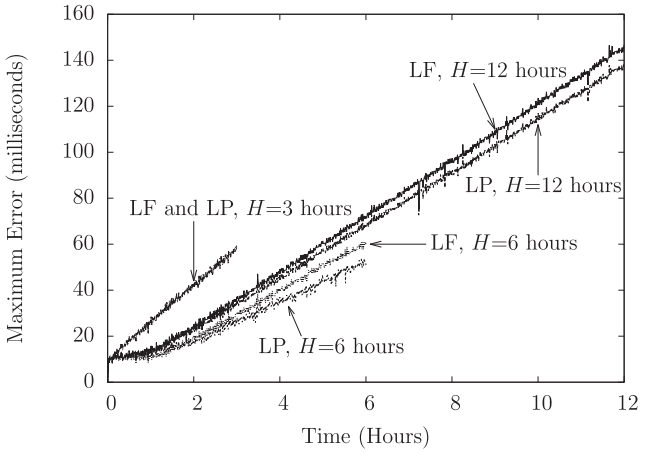


Fig. 6. The maximum offset between the eight Android phones. LF is linear fitting, and LP is linear programming. H is the time between two synchronization instances.

connect to the laptop through this ad hoc network. The round trip delay between the smartphones and the laptop is within 3 milliseconds in low traffic settings. The smartphones calibrate their own clock skews and are scheduled to send a packet at the beginning of every minute. For example, at local time 5 pm $16' 45.500''$, a phone will sleep for 14.500 seconds and send a packet to the laptop at 5 pm $17' 00.00''$. Upon receiving the packets, the laptop saves its local time, based on which the laptop can calculate the clock offset between the eight smartphones.

Fig. 6 shows the maximum clock offset between the eight calibrated smartphones. For both linear programming and linear fitting methods, we set H as 3, 6, and 12 hours. As shown in Fig. 6, when $H = 6$ and 12 hours, the maximum offset between the clocks is always within 100 milliseconds. The standard deviation of the offset of all the phones when $H = 6$ is only 6.66 milliseconds. In addition, we observed that the linear programming method has smaller offset than the linear fitting method when $H = 6$ and 12 hours. When $H = 3$ hours, the performance of the two algorithms is similar. The result shows that both algorithms can predict the clock skew more accurately when $H = 6$ hours than when $H = 12$ hours or $H = 3$ hours. When H is too small (e.g., $H = 3$ hours), the sampling point is not well separated to capture the clock's skew. On the other hand, when H is too large (e.g., $H = 12$ hours), the clock's skew may have changed during the measurement process, which also results in inaccurate predictions. In Fig. 6, the lines are fuzzy because the packet delay from the smartphones to the laptop varies. This experiment shows that the smartphones can be synchronized within 100 milliseconds if they can synchronize with an NTP server every 6 hours.

5.2 Measurement and Estimation of Energy Overhead

To show that the energy overhead of the clock calibration is small, we measured the energy consumption of an HTC G1 with Android OS in different energy states. We took out the battery of the phone and powered the phone with a 4.5-V power supply. A RadioShack 22-812 digital multimeter was used to monitor the current on the power supply cable. The collected data from the digital multimeter were saved to a

TABLE 1
Energy Measurement Summary

States	Power	Std. dev
Idle	$P_{idle} = 0.021$ W	0.0319 W
Screen-on	$P_{scrn} = 0.472$ W	0.0189 W
Ad hoc test	$P_{adhoc} = 10.297$ J/Period	1.192 J/Period
NTP test	$P_{ntp} = 18.299$ J/Period	3.426 J/Period

Linux desktop through an RS232 cable. The maximum sampling rate of the digital multimeter is 1 Hz. We experimented with four different states of the smartphone as listed below:

- *Idle*. Both the screen and the WiFi interface are turned off. The other settings are the same as the default settings of the Android 1.6 OS.
- *Screen-on*. The settings in this state are the same as the settings in the idle state except that the screen is turned on with a constant brightness value.
- *Ad hoc test*. In addition to the screen-on state, an application periodically runs a sequence of tasks. In each period, it turns on the WiFi interface with the ad hoc setting, broadcasts two beacons, and then turns off the WiFi interface. The ad hoc mode of HTC G1 phone is enabled by the binaries provided by the android-wifi-tether project [27]. The average length of a period in this application is $L_{adhoc} = 10.023$ seconds.
- *NTP test*. In addition to the screen-on state, an application periodically runs a sequence of tasks. In each period, it turns on the WiFi interface with the infrastructure setting, connects to a wireless access point, retrieves 10 time stamps from NTP server 0.north-america.pool.ntp.org, runs the linear fitting clock calibration algorithm, and then turns off the WiFi interface. The average length of a period is $L_{ntp} = 18.043$ seconds.

The screen-on state was tested because when the screen is turned off, the Android OS may suspend the test applications in the ad hoc test state and the NTP test state. Later, we will subtract the screen-on state energy in our calculation. L_{adhoc} and L_{ntp} are the shortest time for finishing the tasks in the ad hoc test state and the NTP test state, respectively. The energy of the phone in the four states is measured multiple times and the mean and standard deviations are presented in Table 1. Instead of Watts, J/Period is used as the energy measurement unit in both the ad hoc test state and the NTP test state because in these two states, the energy consumption rate is nonuniform within a time period. The standard deviations for the two states are calculated based on the energy consumption in each period.

We now estimate the energy overhead of calibrating the clock using the measured data. The slot size is set to be the same as a period of the ad hoc state, i.e., $\delta = L_{adhoc}$. In $H = 6$ hours, the average number of wake-up slots is $N_{avg} = dH/\delta$, the idle state energy in 6 hours is $E_{idle} = P_{idle}H$, and

the ad hoc test energy without the screen-on state power is $E_{adhoc} = N_{avg}(P_{adhoc} - P_{scrn}L_{adhoc})$, where P_{adhoc} is the power consumption rate in the ad hoc state as defined in Table 1. Because the NTP test only runs one round every 6 hours in our design, the NTP test energy without the screen-on state power is $E_{ntp} = 1\,(P_{ntp} - P_{scrn}L_{ntp})$. E_{ntp} is the energy overhead for synchronizing the clock of a smartphone. Applying the values in Table 1, $E_{ntp} = 9.783$ J. The relative energy overhead E_o is estimated using

$$E_o = E_{ntp}/(E_{adhoc} + E_{idle}). \quad (13)$$

When $d = 1\%$, (13) shows that the relative overhead $E_o = +1.7\%$. When $d = 5\%$, the relative overhead $E_o = +0.93\%$. In other words, the additional energy overhead required for calibrating the clock is negligible. Since E_{adhoc} increases as d increases, while the energy overhead E_{ntp} does not change, the relative energy overhead E_o decreases as the duty cycle d increases.

6 Simulation and Comparison

In Section 6.1, we will introduce another synchronous neighbor discover protocol called PRS. It is used as a baseline synchronous protocol. Then, in Section 6.2 we simulate and compare RBTP with other five state-of-the-art protocols: Birthday protocol [16], Disco [6], U-Connect [13], Searchlight-S [3], and Searchlight-R [3]. When simulating the Searchlight protocol, we use the stripped version of the protocol. Bakht et al. [3] have shown that the stripped version of the Searchlight protocol performs better than the unstripped version.

6.1 Another Synchronous Scheme: Pseudo-Random Shuffling (PRS)

In this section, we present a simple synchronous neighbor discovery scheme called *Pseudo-Random Shuffling*. PRS serves as a baseline synchronous algorithm to compare with RBTP. We focus on examining the expected contact latency of this scheme.

PRS divides time into constant length time frames and each frame consists of N slots. The idea of PRS is to randomly select n slots from a frame if a node has n wake-up instances in that frame. This is done by randomly shuffling the indices of these N slots and selecting the first n slot indices as wake-up slots. Roughly speaking, the more the selected slots overlap across different nodes, the lower would be the contact latency. The frame number f, which is known by all nodes, is used as the seed of the shuffling algorithm for frame f, where $f = 0, 1, 2, \ldots$. In this way, PRS guarantees that nodes with different duty cycles can overlap to the maximum extent possible, and the scheduled wake-up slots are well distributed within a frame (when analyzing many frames).

Fig. 7 shows the expected contact latency of RBTP and PRS using simulations. In the simulations, the contact latency is represented in terms of the number of slots, and the frame size is 1,024 slots. The simulation result shows that the expected contact latency of PRS is worse than RBTP. This indicates that the specially designed binary partition method in RBTP can better exploit the synchronous clock.

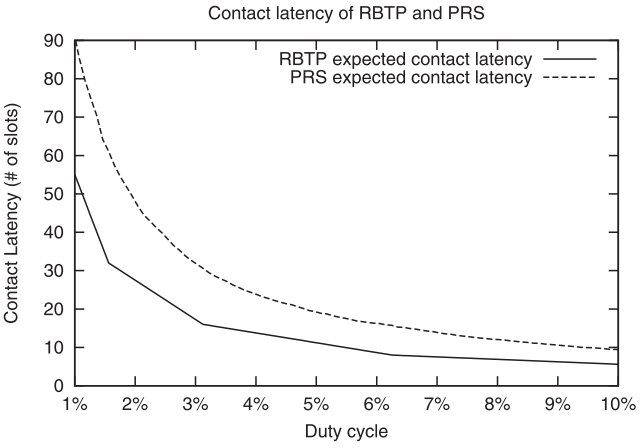


Fig. 7. The expected contact latency of RBTP and PRS in simulation.

6.2 Simulation Based on NS-3 [28]

The NS-3 simulator uses the constant speed propagation delay model [19] and the Friis propagation loss model [20]. Both models are built-in models in NS-3. We assume two phones can discover each other if one phone receives beacon(s) from another phone and immediately sends back a beacon in response. The phones use 100 milliseconds as the slot size. The frame size for RBTP and PRS is always 1,024 slots. For U-Connect, Searchlight-R, and Searchlight-S, the frame size of a node is determined by the node's duty cycle. In all the simulation instances, the offset of a clock is determined by a Gaussian random variable generator with mean 0 and standard deviation 6.66 milliseconds, which is the same as standard deviation of the measured clock offset in Section 5.1 when using linear fitting method with 6 hours as the clock synchronization period.

Fairness is guaranteed by letting all the nodes have the same duty cycle across different protocols. If in a simulation the nodes need to use different duty cycles, the duty cycle of a node is uniformly selected from the range of the specified duty cycles. Although the other protocols besides RBTP and PRS are designed for asynchronous clock, we found that their performance improved or remained unchanged when using synchronous clock (Section 6.2.1).

6.2.1 Contact Latency

We created a scenario with 10 nodes that are within the contact range of each other. The transmission range of the nodes is 75 meters. The nodes move in a 50×50 m^2 region with the random walking mobility model [21]. The discovery rate of a node in slot t is defined as the fraction of nodes discovered out of all its neighboring nodes, in all slots up to and including slot t. Because the nodes form a clique in our setting, the neighboring nodes are always the other nine nodes in any time slot for any node. The expected discovery rate is calculated after each time slot in the simulation. Note that the discovery rate at t slots is the same as the cumulative probability of the contact latency t. The discovery process is simulated for 1,000 rounds for each protocol. In each round of simulation, the start time of a protocol is randomly chosen and each protocol runs for 3,000 time slots. We simulate other protocols with both synchronous clock and asynchronous clock, but we always

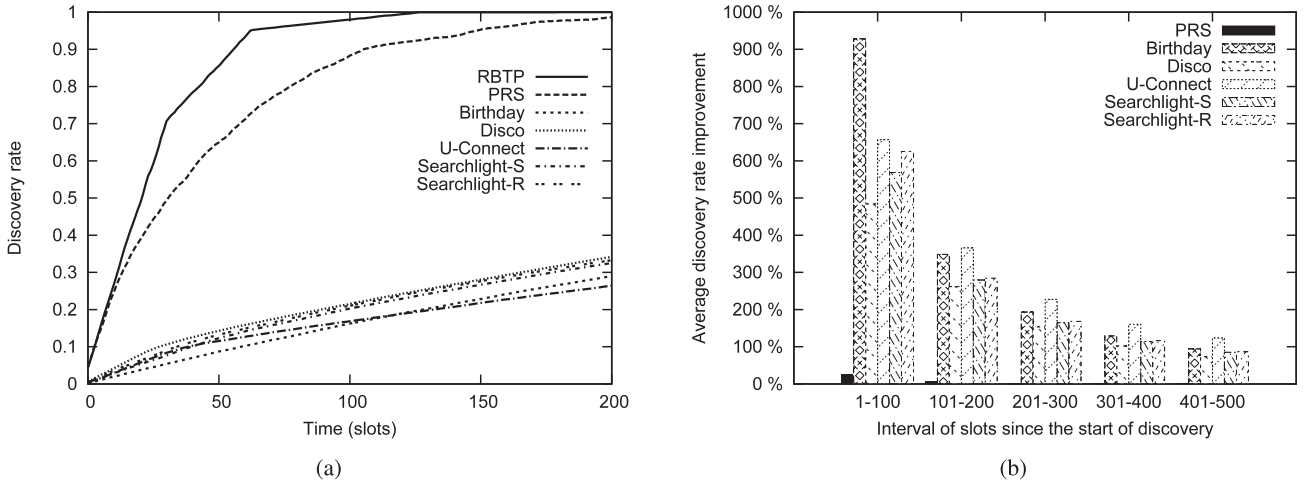


Fig. 8. Low duty cycle (1-5 percent) (a) cumulative discovery latency. (b) improvement of RBTP over other protocols in terms of the average discovery rate in different time periods.

use synchronous clock for RBTP. Two duty cycle settings are used: low duty cycle setting and high duty cycle setting. In the low duty cycle setting, the duty cycles of the nodes are uniformly chosen within the range between 1 and 5 percent. In the high duty cycle setting, the duty cycles of the nodes are uniformly chosen within the range between 5 and 10 percent.

Fig. 8a shows the discovery rate when the protocols use synchronous clock and the low duty cycle setting. Fig. 8b compares the discovery rate of RBTP with the other protocols for different number of elapsed slots. Fig. 8a shows that both RBTP and PRS have significantly higher discovery rates compared to the other protocols. This indicates that RBTP and PRS will have relatively lower contact latency. Fig. 8b shows that in a short period, RBTP has discovered more nodes than other protocols. We observed that the performance of PRS and RBTP is close. The reason that PRS and RBTP perform better than the asynchronous protocols is that they have the maximum number of overlapping wake-up slots s they can exploit synchronized time. In addition, in low duty cycle settings, missing a discovery chance will lead to long discovery latency because the number of wake-up slots is small. The performances of the other asynchronous protocols are similar. Note that different from the simulation results shown in [3], our simulation result does not show Searchlight to be significantly better than Disco and U-Connect. Fig. 9a shows the discovery rate of the protocols when the duty cycles of the nodes are between 5 and 10 percent. The performances of RBTP and PRS are still better than the other asynchronous protocols. However, Fig. 9b shows that the improvement of RBTP is smaller than the improvement in the low duty cycle case shown in Fig. 8b. This is due to the fact that the asynchronous protocols perform better at higher duty cycles. Meanwhile, the performance of RBTP did not improve as significantly as the asynchronous protocols because the discovery rate of RBTP is already high in low duty cycle settings. The trend is that the performance gap of the protocols will be smaller when the protocols use higher duty cycles. For example, if the duty cycle is 50 percent, roughly speaking, all the protocols will wake up the phone every other slot and the contact latency

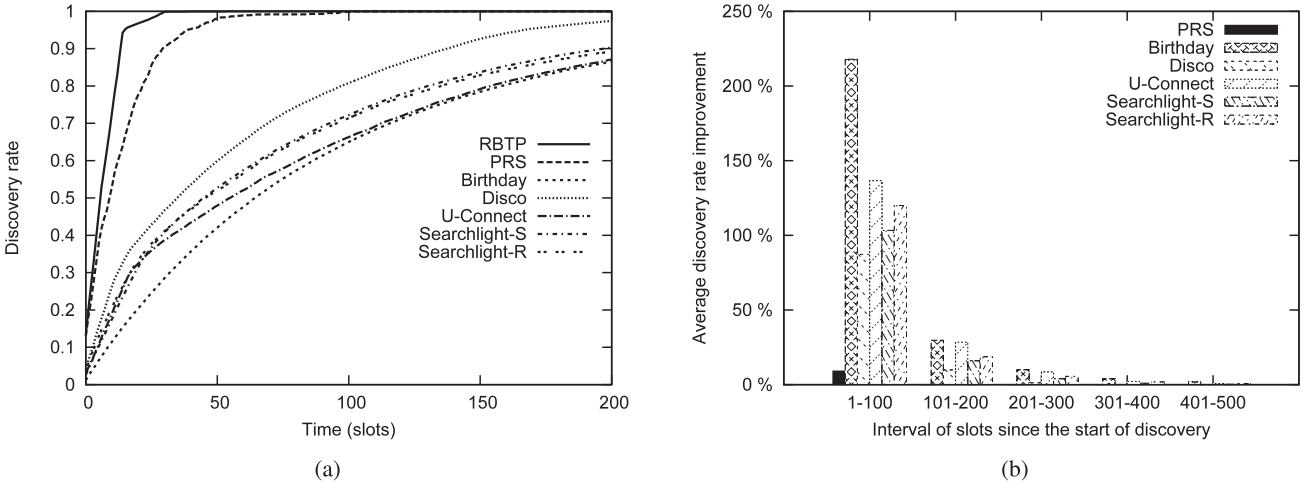


Fig. 9. High duty cycle (5-10 percent) (a) cumulative discovery latency. (b) improvement of RBTP over other protocols in terms of the average discovery rate in different time periods.

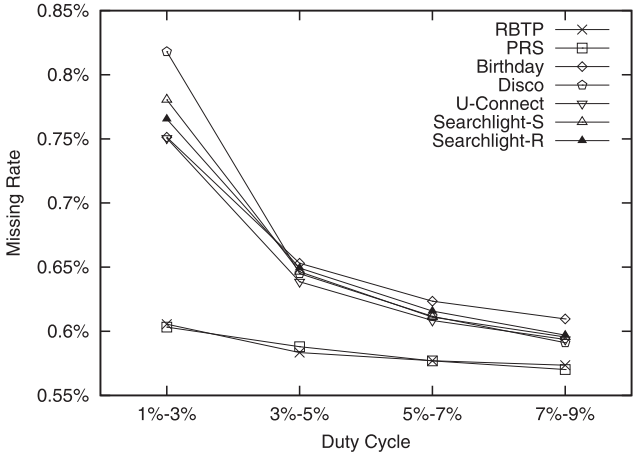


Fig. 10. The missing rate of the protocols for different duty cycle ranges.

will be similar. The simulation results in Figs. 8 and 9 show that RBTP performs significantly better than state-of-the-art asynchronous protocols.

In addition, we observed that if synchronous clock is used in the simulation, the asynchronous protocols except the Birthday protocol show improved performance. For example, when using asynchronous clock with duty cycle between 1 and 10 percent, the expected discovery rate increases by about 2 percent for both Disco and U-Connect, and increases by 1 percent for Searchlight-S and Searchlight-R. We believe the reason is that when using synchronized clock, the wake-up slots of nodes with the same duty cycle are also synchronized, thus these nodes will wake up in the same slots. The performance of the Birthday protocol is not affected because it treats all the time slots uniformly.

6.2.2 Missing Rate

We let 10 nodes move with the random mobility model in an area of size $200 \times 200\ \mathrm{m}^2$. As the transmission range is 75 meters, any pair of nodes will not always be within their mutual contact range when moving. The missing rate is simulated for four different settings of duty cycles. Each node is assigned a duty cycle uniformly chosen within the range in each setting. The simulation duration for each protocol is the same. The result of the simulation is shown in Fig. 10. The horizontal axis of Fig. 10 is the duty cycle range used in each set of simulation.

As the duty cycle increases, the missing rate decreases. The synchronous clock protocols (RBTP and PRS) have similar performance. They both have very low missing rate even when the duty cycle is between 1 and 3 percent. On the other hand, the asynchronous protocols have poor performance in low duty cycle settings.

7 Related Work

Researchers have proposed physical proximity-based applications, such as BlueAware [7], E-SmallTalker [31], and E-Shadow [26]. Eagle and Pentland [7] designed an application that senses users' proximity via Bluetooth. It records the Bluetooth identity of nearby users and queries a database to get their on-file profiles. E-SmallTalker and E-Shadow are similar applications. Teng et al. [26] designed an application that can publish news and messages to users in physical proximity. These applications exploit the capabilities of the proximity network to establish contact and share messages with nearby devices. Bluetooth is used in [7], [30], [31], and both Bluetooth and WiFi are used in [26]. The energy consumption rate of Bluetooth is lower than WiFi, but the communication range is only about 10 meters that limits its capability. WiFi offers longer distance, however, it consumes more energy than Bluetooth.

Research on MAC layer technologies has also produced several schemes to reduce energy cost for discovering neighboring sensors and phones. Energy conserving MAC protocols in wireless sensor networks, such as S-MAC [32] and B-MAC [23], assume that the neighbor nodes are in the vicinity of each other, so the sensors can exchange schedules or send long preambles to synchronize with neighbors. A neighbor discovery protocol for static wireless sensor networks [8] has focused on minimizing collisions and discovering all the neighbors. In our scenario, the neighbors may arrive in the transmission range at arbitrary times, which makes the existing protocols inapplicable. SSCH [2] is a scheme that allows multiple wireless devices to utilize multiple wireless channels. It focuses on increasing the capacity rather than minimizing energy consumption. Wang et al. [30] show that if both the contact duration and the intercontact duration are stationary, the missing contact probability can be minimized.

Current neighbor discovery schemes for mobile nodes mainly use a slotted time model [29], [16], [6], [13], [3]. In a slotted time model, an equal sized slot is the unit of time. A device can either wake up in a slot or sleep in a slot. The mutual discovery between two devices happens when two devices are both awake in the same slot and they are within the transmission range of each other. When the clock is not synchronized, the slots may be unaligned. To address this problem, a common technique is to send beacons at the beginning and the end of a wake-up slot [6], [13], [3]. The schemes using slotted time model can be classified into probabilistic schemes [16] and deterministic schemes [6]. Because all these protocols assume that the clocks on the devices are not synchronized, they can be categorized as asynchronous protocols.

The Birthday protocol [16] is inspired by the "Birthday paradox." Each time slot in the Birthday protocol is determined to be a wake-up slot according to a probability. The drawback of this protocol is that it cannot guarantee the worst-case discovery latency. In deterministic protocols [29], [6], [13], the worst-case discovery latency has an upper bound. Tseng et al. [29] divided the time into time blocks. Each time block has m^2 contiguous slots, and the slots are arranged into a two-dimensional $m \times m$ matrix. Each mobile device can independently choose a row and a column of slots to wake up. The duty cycle is $(2m-1)/m^2$. This scheme guarantees that two devices will have at least two overlapping wake-up slots in every m^2 slots. However, the scheme is not flexible because all the devices have to use the same duty cycle. Dutta and Culler [6] designed a scheme based on the property of prime numbers. In their scheme, a device chooses a set of prime numbers based on its energy budget. When the slot number can be divided by any one of

the prime numbers the user has chosen, the device will wake up in that slot. The device will remain in sleep mode in other slots. For example, if a device chooses prime numbers 3 and 5, the first five slots it will wake up are slots 3, 5, 6, 9, and 10. The duty cycle is $1/3 + 1/5$. The idea behind this scheme is the Chinese Remainder Theorem. U-Connect [13] uses a similar idea, except that it uses one prime number. In U-Connect, if a device chooses prime number p, it will only wake up if the slot number can be divided by p, or the slot number modulo p^2 is less than $(p+1)/2$. So, it wakes up every p slots and keeps in the wake-up state for $(p+1)/2$ slots in every p^2 slots. However, both the average case and worst-case discovery latencies are still large when a device's duty cycle is very low. For example, if two devices both choose prime number 101, the worst-case discovery latency is 10,201 slots. The duration is unacceptable if the slot size is 1 second as in [3]. Searchlight [3] is also a slot-based asynchronous discovery scheme. In a period containing t contiguous slots, Searchlight will wake up the node in two slots: the anchor slot and the probe slot. The anchor slot is the first slot. There are two variants based on the probe slot. In Searchlight-S, in each period the probe slot is selected from the set $\{1, 2, \ldots, \lfloor t/2 \rfloor\}$ in order. In Searchlight-R, the probe slot is randomly selected from the set $\{1, 2, \ldots, \lfloor t/2 \rfloor\}$. In the stripped versions of the Searchlight-S and Searchlight-R protocols, the probe slots are selected from the set $\{2, 4, \ldots, 2\lceil \lfloor t/2 \rfloor /2 \rceil\}$, i.e., the even slots.

Clock synchronization is considered to be difficult to achieve in previous works [16], [6], [13]. It is also a challenging problem in wireless sensor networks [25]. However, it is relatively more convenient to synchronize the clock for smartphones. Current smartphones have a variety of means to synchronize their internal clocks. Kohno et al. [14] found that the clock skew in different computers can vary between -300 and 300 ppm. However, the skew of a computer's clock is stable. After measuring a clock's skew at different times, the maximum and minimum skew estimate was found to be only 0.67 ppm, which is 0.67 microseconds per second [14]. It indicates that we can design a clock synchronization scheme by estimating the skew of the clock and correcting for it. Moon et al. [18] provide a linear programming method to estimate the skew of a clock.

In this work, we show that current smartphones can take advantage of partially synchronized time for the neighbor discovery process. It can shorten the contact latency for proximity-based discovery applications, and reduce the missing rate. The protocol has both low average case and worst-case contact latencies compared to asynchronous protocols.

8 Discussion

In this section, we summarize the challenges in implementing the neighbor discovery application on smartphones, and discuss some possible future directions for research:

- Although we have shown that the smartphones can calibrate the clocks using the NTP protocol, but if a smartphone cannot synchronize its clock for a long time for reasons such as failure to establish connection with the NTP server, it is possible that the clock's offset exceeds 100 milliseconds. To address this problem, we are planning a future extension of RBTP that allows the smartphones to estimate the accuracy of their clocks and then adapt the slot size to accommodate for the larger offset.
- There are several implementational challenges related to neighbor discovery using smartphones. The neighbor discovery application requires the WiFi module to work in the ad hoc mode. Some devices may not fully support the ad hoc or the power saving modes. For example, we have seen that Android 1.6 OS does not officially support the ad hoc mode. For a smartphone that supports the ad hoc mode, the neighbor discovery application should not interrupt the phone's connection with the WiFi access point. One possible solution is that when the WiFi works in the infrastructure mode, the phone can attempt discovering neighbors connected with the same AP. MultiNet [5] allows simultaneous connections to multiple networks by virtualizing a single wireless card. But it is unknown whether it can be implemented on smartphones.
- Although the clock synchronization requirement can be met on current smartphones, it is still difficult for an application to control the WiFi driver in real time. Many works on designing IEEE 802.11 power management schemes are based on simulations [29], [24], [1] rather than real implementations. Bakht et al. [3] implemented their neighbor discovery protocol on smartphones; however, the switching of the WiFi mode takes around 1 seconds, which is not practical. On the other hand, as discussed in the Section 1, these engineering problems can be easily solved by the manufactures.
- The IEEE 802.11 protocol allows devices to use any one of the 13 channels. When smartphones use different channels, the neighbor discovery problem would be more challenging.

Besides these challenges, there are many interesting and practical questions related to this research, and we leave them as our future work. One natural question is that with this application, a user can get a log of records indicating who she or he has met when and where. It has the potential to enable many unique mobile-enabled applications that will define the future. How to utilize this information is an open question.

Another interesting direction to this problem can be using time, location, and motion-based discovery schedule to discover neighbors. For example, when the user is present at some location of interest, at certain time of day, or when walking, the discovery service automatically starts scanning for neighboring users. Knowledge of the user's habits can be exploited to improve the design further.

9 Conclusion

The neighbor discovery problem in mobile wireless networks is fundamental for applications such as mobile advertisements, and proximity-based social networking. User mobility and energy constraints pose challenges in solving this problem. To support these emerging applications and minimize both the contact latency and the missing

rate, we proposed a neighbor discovery scheme. Our proposed solution, RBTP has bounded performance with respect to an optimum solution in terms of expected and the worst-case contact latencies. We observed that significant reduction in latency can be achieved by leveraging time synchronization. Our experiment shows that clock synchronization requirement of RBTP is achievable and its energy overhead is negligible. The simulations show that the performance of RBTP is significantly better than state-of-the-art asynchronous protocols.

REFERENCES

[1] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 Power-Saving Mode for Mobile Computing in Wi-Fi Hotspots: Limitations, Enhancements and Open Issues," *Wireless Networks,* vol. 14, no. 6, pp. 745-768, 2008.

[2] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," *Proc. ACM MobiCom,* pp. 216-230, 2004.

[3] M. Bakht, M. Trower, and R.H. Kravets, "Searchlight: Won't You Be My Neighbor?" *Proc. ACM MobiCom,* pp. 185-196, 2012.

[4] F. Ben Abdesslem, A. Phillips, and T. Henderson, "Less Is More: Energy-Efficient Mobile Sensing with Senseless," *Proc. First ACM Workshop Networking, Systems, and Applications for Mobile Handhelds,* pp. 61-62, 2009.

[5] R. Chandra and B.P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," *Proc. IEEE INFOCOM,* vol. 2, pp. 882-893, 2004.

[6] P. Dutta and D. Culler, "Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications," *Proc. Sixth ACM Conf. Embedded Network Sensor Systems (SenSys '08),* pp. 71-84, 2008.

[7] N. Eagle and A. Pentland, "Social Serendipity: Mobilizing Social Software," *IEEE Pervasive Computing,* vol. 4, no. 2, pp. 28-34, Jan.-Mar. 2005.

[8] S. Vasudevan et al., "Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem," *Proc. ACM MobiCom,* pp. 181-192, 2009.

[9] Facebook, http://www.facebook.com, June 2012.

[10] HTC G1, http://en.wikipedia.org/wiki/HTC_Dream, June 2012.

[11] M. Gast, *802.11 Wireless Networks: The Definitive Guide,* chapter 8, pp. 188-196. O'Reilly Media, 2005.

[12] D.C. Jefferson, S.M. Lichten, and L.E. Young, "A Test of Precision GPS Clock Synchronization," *Proc. IEEE Int'l Frequency Control Symp.,* pp. 1206-1210, 1996.

[13] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-Connect: A Low-Latency Energy-Efficient Asynchronous Neighbor Discovery Protocol," *Proc. ACM/IEEE Ninth Int'l Conf. Information Processing in Sensor Networks (IPSN '10),* pp. 350-361, 2010.

[14] T. Kohno, A. Broido, and K.C. Claffy, "Remote Physical Device Fingerprinting," *IEEE Trans. Dependable Secure Computing,* vol. 2, no. 2, pp. 93-108, Apr. 2005.

[15] LinkedIn, http://www.linkedin.com, June 2012.

[16] M.J. McGlynn and S.A. Borbash, "Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks," *Proc. ACM MobiHoc,* pp. 137-145, 2001.

[17] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," IETF RFC 1305, 1992.

[18] S.B. Moon, P. Skelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements," *Proc. IEEE INFOCOM,* vol. 1, pp. 227-234, 1999.

[19] "ns3::ConstantSpeedPropagationDelayModel Class Reference," http://www.nsnam.org/doxygen-release/classns3_1_1_constant_speed_propagation_delay_model.html, June 2012.

[20] "ns3::FriisPropagationLossModel Class Reference,"http://www.nsnam.org/doxygen-release/classns3_1_1_friis_propagation_ loss_model.html, June 2012.

[21] "ns3::RandomWalk2dMobilityModel Class Reference,"http://www.nsnam.org/doxygen/classns3_1_1_random_walk2d_mobility_model.html, June 2012.

[22] J. Paek, J. Kim, and R. Govindan, "Energy-Efficient Rate-Adaptive GPS-Based Positioning for Smartphones," *Proc. ACM MobiSys,* pp. 299-314, 2010.

[23] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. ACM SenSys,* pp. 95-107, 2004.

[24] D. Qiao and K.G. Shin, "Smart Power-Saving Mode for IEEE 802.11 Wireless LANs," *Proc. IEEE INFOCOM,* vol. 3, pp. 1573-1583, 2005.

[25] B. Sundararaman, U. Buy, and A.D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks: A Survey," *Ad Hoc Networks,* vol. 3, no. 3, pp. 281-323, 2005.

[26] J. Teng, B. Zhang, X. Li, and X. Bai, "E-Shadow: Lubricating Social Interaction using Mobile Phones," *Proc. IEEE 31st Int'l Conf. Distributed Computing Systems (ICDCS),* 2011.

[27] "android-wifi-tether," http://code.google.com/p/android-wifi-tether, June 2012.

[28] "NS-3 Network Simulator," http://www.nsnam.org, June 2012.

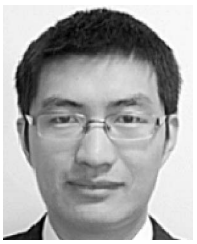
[29] Y.C. Tseng, C.S. Hsu, and T.Y. Hsieh, "Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks," *Proc. IEEE INFOCOM,* vol. 1, pp. 200-209, 2002.

[30] W. Wang, V. Srinivasan, and M. Motani, "Adaptive Contact Probing Mechanisms for Delay Tolerant Applications," *Proc. ACM MobiCom,* pp. 230-241, 2007.

[31] Z. Yang, B. Zhang, J. Dai, A.C. Champion, D. Xuan, and D. Li, "E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity," *Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 468-477, 2010.

[32] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1567-1576, 2002.

[33] B. Zhang, J. Teng, X. Bai, Z. Yang, and D. Xuan, "P3-Coupon: A Probabilistic System for Prompt and Privacy-Preserving Electronic Coupon Distribution," *Proc. IEEE Int'l Conf. Pervasive Computing Comm. (PerCom),* pp. 93-101, 2011.



Dong Li received the BS degree in information security from the University of Science and Technology of China, Hefei, in 2009. He is currently working toward the PhD degree at the Ohio State University, Columbus. His research interests include wireless networks and mobile device energy management. He is a student member of the IEEE.

Prasun Sinha received the BTech degree from the Indian Institute of Technology Delhi in 1995, the MS degree from Michigan State University in 1997, and the PhD degree from the University of Illinois, Urbana-Champaign in 2001. He is currently an associate professor in the Department of Computer Science and Engineering at the Ohio State University. From 2001 to 2003, he was a member of the technical staff at Bell Labs in Holmdel, New Jersey. His research focuses on ubiquitous networking. He is on the steering committee for IWQoS and has served (or is currently serving) as a TPC chair/cochair for ICDCN 2013, ACM/IEEE IWQoS 2011, ICST BROADNETS 2010, and ICST QShine 2009. He is currently serving on the editorial board of the *IEEE Transactions on Mobile Computing* and has served on the editorial board of *Transformative Works and Cultures* in the past. He has received several awards including the Lumley Research Award (OSU, 2009), CAREER award (US National Science Foundation, 2006), Ray Ozzie Fellowship (UIUC, 2000), Mavis Memorial Scholarship (UIUC, 1999), and Distinguished Academic Achievement Award (MSU, 1997). He is a senior member of the IEEE.