This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

The sleepy bird catches more worms: revisiting energy efficient neighbor discovery

Lorenzo Bracciale, Pierpaolo Loreti, Giuseppe Bianchi
University of Rome "Tor Vergata" Rome, Italy
{lorenzo.bracciale,pierpaolo.loreti,giuseppe.bianchi}@uniroma2.it

Abstract—Neighbor discovery is a primary enabling ability for many emerging mobile applications. Due to its significant impact on the energy budget of battery equipped devices, energy preserving solutions have been investigated for a long time, often introducing duty cycling. Ultimately, these solutions settle for trade-offs between energy and performance, leaving the final decision on how much energy to allocate to neighbor discovery in the hands of application developers or system engineers. In other words, someone must decide if an improvement in the quality of the discovery (e.g. better discovery latency) is worth an increase in energy consumption. In this paper, we devise a different approach in order to answer the following basic question: how many contacts can a smartphone discover using its battery energy budget?
The answer clearly depends on the adopted discovery algorithm, on the mobility conditions and on the stochastic characteristics of the encounters. However, we demonstrate that there is a natural optimum duty cycle that maximizes the number of discovered contacts in almost every real application scenario. This optimum is natural in the sense that it does not depend on any system level parameter or performance requirements. It depends uniquely on the stochastic characteristics of the meeting process between the nodes. We present an analytic analysis and devise a practical algorithm that dynamically adapts the duty cycle length to the time-varying context, without the need to make assumptions on (or predict) the distribution of the contact duration. The findings presented in the paper are validated against data coming from real human mobility traces and implemented on a mobile application.

Index Terms—Neighbor Discovery, Opportunistic Networks, Energy Saving, Mobile Network, Android, Duty Cycle, Loose synchronization

1 Introduction

Using their commodity short range wireless interfaces, such as Wi-Fi or Bluetooth, smartphones can communicate with each other and with other devices in their proximity, realizing so-called opportunistic mobile networks or enabling location based services such as [1]. Despite the increasing interest in proximity communications due to the absence of any infrastructure requirements, there are several limiting factors that prevent a real diffusion of this technology. Among these is the high energy cost needed to discover neighbor nodes [2].

The neighbor discovery problem has been investigated for a long time, both in sensor and mobile networks. In particular, in this work we focus on discovering *new* contacts, i.e. discovering nodes for the first time. In fact, after a pair of nodes has discovered each other they can schedule periodic communication (re-discovery) through a rendez-vous protocol such as [3]. The problem basically consists in finding neighbors reachable though a short range wireless technology (e.g. Wi-Fi, Bluetooth or IEEE 802.15.4) in a way that is efficient in terms of some metrics such as discovery latency, missing probability and power consumption.

Yet devising a good and energy efficient discovery protocol is not an easy task and the main reason is very basic: discovering neighbors requires polling the environment but *polling for rare events has never been a good choice*, especially in terms of energy efficiency. Indeed, periodically checking for the presence of new nodes can be an extremely energy consuming operation. Moreover, the result is useless most of the time because "interesting" encounters (e.g. another node with the same app installed) might take place very rarely.

As pointed out in several works [4], keeping the radio interface active to catch discovery probes, greatly impacts the energy budget of battery-powered devices.

To overcome this limitation, a popular approach involves the introduction of a duty cycle, so that a mobile phone's radio interface can be activated for a short period of time to probe for neighbor devices, and then be switched off for a relatively long period to save energy. The rationale of putting the radio interface to sleep is clear if we look at the statistics of contact duration in real world cases (for instance by analyzing some real world traces coming from the MIT reality mining [5] or from the HAGGLE project [6]). Here, we find that the relevant parameters time scale (e.g. meeting duration, inter-meeting time) are far greater than the actual communication needs for discovery.

This technique is usually difficult to optimize because most encounters are almost unpredictable. Indeed, in general, nodes move inside environments whose statistics are unknown or only partially known (either in terms of statistical parameters or in terms of the overall distribution involved) and vary dramatically over time.

Natural optimization

In literature, several trade-offs emerge if we consider on the one hand the energy consumed for neighbor discovery and on the other all related performance metrics such as the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

probability of missing a contact [7] [8] [4], or the discovery latency [9]. According to this model of operation, the choice of the right amount of energy to dedicate to the discovery process is handed on to the application or system developers. In this paper we tackle the same problem under a new perspective. We investigate if *it is worth* using a given amount of energy, in order to *discover the greatest number of contacts with a fixed energy budget* (e.g. one battery charge of a commercial smartphone).

From this new perspective there arises a natural trade-off and a consequent optimum point, which, so far, to the best of our knowledge, is undiscovered. "Natural" in the sense that if we consider the case of "always on" (when the radio interface is always in the active state), the absolute number of contacts that we can discover will be severely limited by the duration of our battery which would last only a few hours due to this extensive usage. On the other hand, if we consider the opposite "always off" case (when the radio interface is always turned off), our battery lasts several hours but as a matter of fact we will not be able to discover any contacts. By introducing a duty cycle we can move between these two borderline cases and possibly find an optimum value with which we can discover the maximum number of contacts for a given energy budget. As will be explained later on in this work, in practical cases this often happens because contact duration follows a "human time scale" (order of seconds) so nodes can sleep for significant periods of time [10] [11].

We point out that maximizing the total number of discoverable contacts with a battery charge is particularly relevant in several practical cases such as the many (cumbersome) delay tolerant applications. An example of these applications is the opportunistic file sharing service since, given that every node has the same probability of having interesting data to share, more is better. In these scenarios, the extremely long periods needed to provide the service and the lack of synchronous interaction with users (unattended mode), can lead to neglecting any optimization of discovery times and of the probability of missing a *specific* contact. At the very end, what really matters is only the overall number of different nodes encountered before the mobile phone battery runs out of power.

Contributions

The main contributions of this work are the following:

- we formalize the problem of the maximum number of discoverable peers with an energy budget.
- we quantify the performance of a duty cycle based discovery protocol using the default synchronization capability of modern smartphones.
- we analyze the problem under different mobility hypotheses, investigating the presence of an optimum point. Then we provide a closed form solution for some mobility patterns and validate it using data coming from real mobility traces.
- we design a simple algorithm that exploits our theoretical findings to dynamically adapt the duty cycle to changing environments. All this with no assumption on the distribution of contact duration times, which is the main statistical distribution involved in the optimization.
- we demonstrate the feasibility of the solution through an open source application for smartphones.

The paper is organized as follows. Section 2 presents the reference scenario. Section 3 presents a novel theoretical approach to optimize the duty cycle to discover the maximum amount of contacts with an energy budget. In section 4 we propose a distributed adaptive algorithm called "CATNAP" and in section 5 we assess the related performance using models based on empirical human mobility traces. In section 6 we present a proof-of-concept implementation on Android devices. A review of the related works is reported in section 7. Finally, conclusions are drawn.

2 System Model

2.1 Synchronous duty cycle

Let us assume that all nodes in the network are synchronized and that they periodically turn their wireless interfaces on and off. During the on period all the devices can discover their neighbors by sending and receiving presence data packets. Each node stays on for T_{on} seconds for each cycle. During the off period, a node can neither transmit nor receive packets. A node remains in this state for T_{off} seconds. This behavior is represented in figure 1.

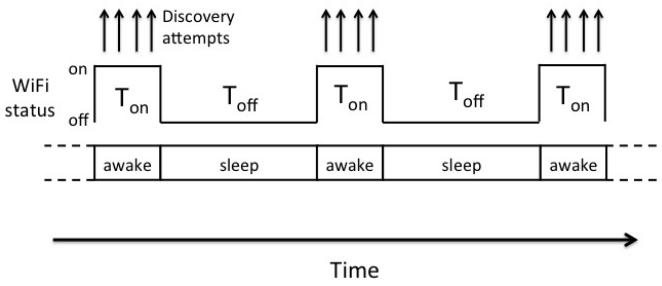


Fig. 1. Each node periodically turns the Wi-Fi interface on and off, implementing a duty cycle to save energy, with *human time-scale*

The system period is then $T = T_{on} + T_{off}$, hence the duty cycle is $d = T_{on}/T$.

Generally speaking, each node can have its own values of T_{on} and T_{off}. We will discuss this case in section 4. In this section and in the next, for the sake of simplicity, we will assume that every node has the same values of T_{on} and T_{off} and that they are synchronized with one another.

We point out that the global synchronization hypothesis is far from being unrealistic. Indeed, by default, modern smartphones periodically make use of NTP to synchronize against a reference time server[1]. This brings about a global *coarse grain* synchronization. A more in-depth discussion on this topic can be found in [11] and [12].

1. From Android 4.0 on, the time update via NTP (or NITZ) is implemented natively by the service "Network Time Update". In addition, by analyzing the source code of Android, we can see that the resync is performed every 24 hours. https://github.com/android

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

2.2 The duty cycle gain

Before going any further, we want to briefly demonstrate the extent of the gain of introducing a duty cycle approach by measuring the battery duration on real smartphones.

If we focus on Wi-Fi, we find many literature studies that evaluate the energy consumption accountable to the different parts of the stack (e.g. [13]). However, from a practical point of view, the real impact of the power consumption of the radio interface on a smartphone battery is difficult to assess as it varies dramatically with hardware (phone, battery), software (e.g. vendor specific rom/patches) and time (batteries are subject to wear and tear).

For this reason we resort to a measurement technique that can be included in custom applications [2]. The proposed measurement methodology leverages the battery level provided by the operating system which clearly does not provide the same precision of a professional measurement tool such as a power analyzer.

We evaluated the battery drain process of some Android smartphones. A custom *app*, every period T: i) turns on the Wi-Fi interface; ii) configures the Wi-Fi interface in ad-hoc mode; iii) sends one UDP packet per second for $T_{on} = 5s$ containing the battery level; iv) sleeps for $T_{off} = T - T_{on}$. We kept the phone in standby mode with the screen turned off without activating any power locks during the OFF time (neither the CPU lock nor the Wi-Fi interface lock).

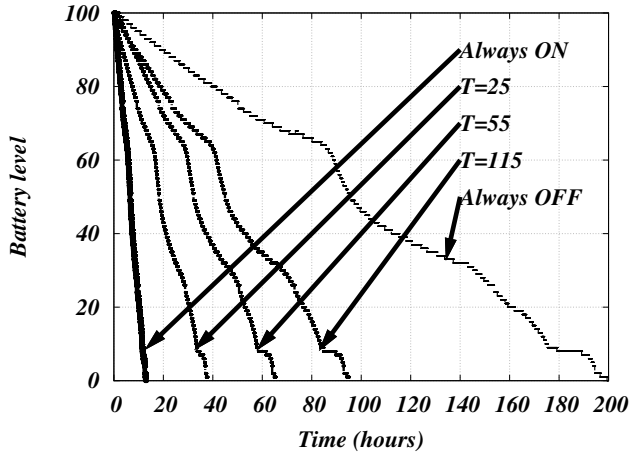


Fig. 2. Battery drain process while keeping the wireless interface always on, always off, or modulating between T=25s, T=55s, T=115s

Figure 2 presents the results of the battery duration of an Android Galaxy Nexus for the following cases: i) *always on*: The Wi-Fi radio interface is always kept on. This is our reference lower bound for battery duration time. ii) *always off*: The Wi-Fi radio interface is always kept off. This is our reference upper bound for battery duration time (in this case the battery level is logged in a local file). iii) T_{off}=15s,25s,55s,115s.

Figure 2 shows that even with a high duty cycle of 20% (T=25s), battery duration is doubled if compared to the *always on* case. In particular, from experiments we confirm that what really makes the battery last longer is the amount of time the phone is in sleeping state, whereas, for instance, the amount of data transmitted when the phones are in the ON state does not significantly impact energy consumption [3].

For this reason we model the battery drain process with two coefficients α and β that represent the percentage of battery drain per second during, respectively, the ON and the OFF states.

For instance, in the case reported in figure 2 we have $\alpha = 2.17 \times 10^{-3}$ and $\beta = 1.39 \times 10^{-4}$ (percentage/second).

Therefore, we can express power consumption during a cycle T as $\mathcal{E}(T) = \alpha T_{on} + \beta T_{off}$.

2.3 Problem statement

We want to set the value of T_{on} and T_{off} to maximize the number of new contacts discovered using the synchronous discovery protocol described so far.

Despite the apparent technological simplicity, the tuning of these parameters is far from trivial:

T_{on} : Given that synchronization between nodes is not perfect, node wake-up times could be slightly shifted. For this reason, T_{on} should be set long enough to allow different mobile nodes to stay awake at the same time. This depends on mobile phone clock drifts, on the precision of the synchronization mechanism [4] and on the time needed to activate the radio interface. As discussed in [11] [12], T_{on} can be set to a constant value e.g. 5s on current smartphones, also considering the time needed to switch the radio interface on and off.

T_{off} : Properly setting the sleeping time is slightly more tricky since it involves a trade-off between the performance of discovery and the amount of energy saved. Let us consider the special case in which all the encounters last more than T_{off}. In this case, the sleeping time introduces only a gain because it increments the battery duration of the device without influencing the number of discovered contacts. In general, the number of missed contacts due to the duty cycle depends on the statistic of the contact duration: increasing the value of T_{off} generally corresponds to a higher probability of missing short contacts. At the same time, as shown in figure 2, long T_{off} saves a significant share of the smartphone energy budget, allowing a longer duration of the discovery process.

Which is better: a longer discovery process with high contact loss probability or a shorter discovery process with low contact loss probability?

The answer clearly depends on contact statistics as will be formalized in the next section.

2. The app's source code is here https://github.com/netgroup/dtn-energy

3. This obviously holds only for small data-rates; neighbor discovery is typically interested in sending small presence packets once in a while.

4. For the sake of this work, the availability of better synchronization between nodes, for instance by using GPS or other network synchronization, allows smaller T_{on} values that in turn results in increased energy saving

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

3 THE ANALYTICAL MODEL

Our goal is to *find the optimum cycle period T that maximizes the number of detected unique contacts with a given energy budget*.

3.1 Assumptions

We assume that the meeting process between nodes is characterized as follows:

- inter-meeting time between each pair of nodes follows an exponential distribution (Poisson process) with rate λ;
- the meeting duration follows a generic distribution $F_D(t)$;
- for ease of presentation and without loss of generality, we consider the discovery process to be instantaneous; the duration of a cycle is T, the power consumption during the ON phase is constant and equal to $\hat{\alpha}$, while power consumption during the OFF phase is $\beta \times T$.

We point out that literature studies such as [14], [15], [16] and [17], proved that inter-contact time distribution between a pair of nodes can be reasonably assumed to be exponentially distributed, if we consider human mobility. In particular, [14] analyzed several popular real world traces showing that 85% of pair distributions fit an exponential law according to χ^2 test. in addition, both [15] and [16] demonstrated that this is not in contrast with the well known heavy tailed distribution (with or without the exponential cut-off) of the *aggregated* inter-meeting time. In fact, a Pareto distribution can be achieved by aggregating several exponential distributions weighted by their rates (i.e. the reciprocal of the averages).

3.2 The renewal process

According to these hypotheses we model the point of view of one node as a $M/G/\infty$ queue. Other nodes meet that node with a λ rate and the meeting duration is a random variable D distributed according to a generic distribution function $F_D(t)$.

We define $n(t)$ as the number of ongoing meetings at time t, assuming $n(t) = 0$ for $t \leq 0$, i.e. assuming an initially empty system, as represented in figure 3.

Proposition 1: In a initially empty $M/G/\infty$ queue system, with λ as the average rate of arrival and $F_D(t)$ as the cdf of waiting time, the average number of users in the system at time T is:

$$E[n(T)] = \lambda \int_0^T [1 - F_D(\tau)]\, d\tau \tag{1}$$

Proof: Let us consider a small interval $\Delta\tau$ at distance $T - \tau$ from the origin, as depicted in figure 4. For the Markovian assumption, during $\Delta\tau$ only one new user can enter the system and this event occurs with probability $\lambda\Delta\tau$. The probability that this user is still in the system at time T is $1 - F_D(\tau)$ and thus the contribution to $n(T)$ of an interval $\Delta\tau$ is 1 with probability $\Delta\tau\,[1 - F_D(\tau)]$ and 0 otherwise. The average is thus $\Delta\tau\,[1 - F_T(\tau)]$. If we repeat this operation on infinitesimal intervals $\Delta\tau \to \delta\tau$ in $[0, T]$, we obtain formula 1. □

Proposition 2: A $M/G/\infty$ queue system, with λ as the average rate of arrival and $F_D(x)$ the cdf of waiting time, that is sampled every T time, the average number of new users at any given time is specified by Equation 1.

Proof: Given that we are interested in new users entering the system, every sample time we need to retrieve all discovered users from the queue. Thus we can consider the truncated contact distribution $F_D^*(x)$ defined as $F_D(x)$ for $x \leq T$ and $F_D^*(x) = 1$ for $x > T$ (i.e. no contact can last more that T). Considering the memoryless property of the arrival process, we can restrict the analysis to the interval $[0, T]$ while it holds for $[0 - \infty]$ because of the periodicity of the system.

Indeed, in our case we are interested in the transitory part and, in particular, the contact time process is renewed every period T as shown in figure 3.

In general, note that as $T \to \infty$ we obtain the famous Little Result.

$$E[n] = \lambda \int_0^\infty [1 - F_D(\tau)]\, d\tau = \lambda E[D]$$

□

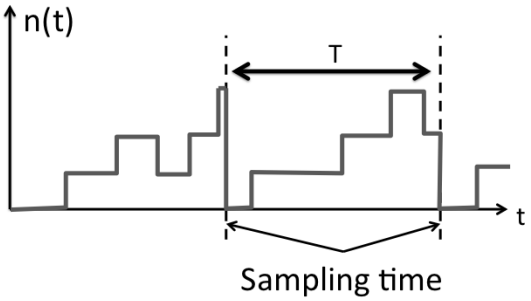


Fig. 3. Representation of the renewal process

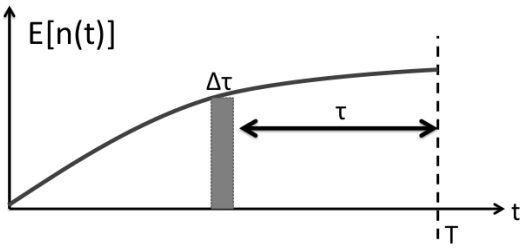


Fig. 4. Representation of the methodology adopted to calculate the average number of users over a period T

3.3 The optimum point

We can formalize the optimum value of the duty cycle that matches our optimization goal under the above-mentioned hypotheses as:

$$\arg\max_T \frac{E[n(T)]}{\hat{\alpha} + T\beta} \tag{2}$$

where $E[n(T)]$ is given by Eq. 1.

Thus, the optimum point is given by the solution of the following differential equation:

$$\frac{\partial}{\partial T} \frac{\lambda \int_0^T [1 - F_D(\tau)]\, d\tau}{\hat{\alpha} + T\beta} = 0 \tag{3}$$

whose solution is $n(T^*) = C(\hat{\alpha} + T^*\beta)$ for $C \in \mathcal{Z}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

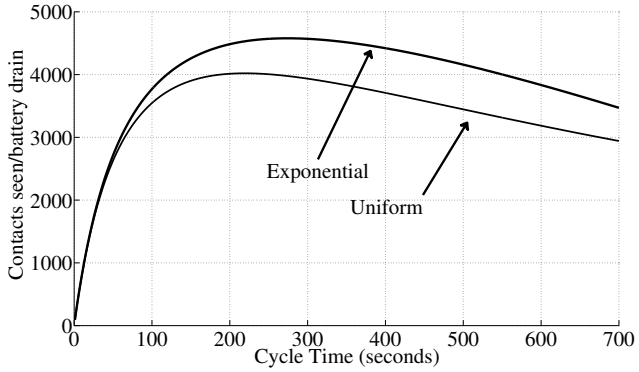


Fig. 5. Optimum in well known distributions: uniform distribution with $\mu = 1/378s^{-1}$ (optimum for $T = 274s$) exponential distribution with parameters $\mu = 1/378s^{-1}$ (optimum for $T = 219.3s$)

We note that:

- the optimum point *is independent of* λ.
- since both $E[n(t)]$ and $\hat{\alpha} + t\beta$ are two monotonic increasing functions, *there is exactly one optimum point for every environment*

Definition 1: We define an *environment* as a physical place where the nodes move, characterized by a inter new contact time distribution and a contact duration distribution.

The discussion regarding the right model to describe human mobility is still open as it greatly depends on the considered scenario [18] [19] [20]. Below, we analyze the case where the node contact time follows some well known distributions as well as the case where it follows the distribution inferred from some real human mobility traces. We point out that the analysis reported in this section still remains valid for *any* contact duration distribution considered.

Uniform distribution

If we consider the node contact time as uniformly distributed in the interval $[0, 2/\mu]$:

$$F_D(t) = \begin{cases} 0, & t \leq 0 \\ \frac{\mu t}{2}, & \text{if } 0 \leq t \leq 1 \\ 1, & t \geq 0 \end{cases}$$

then, by applying eq 3, the optimum point is:

$$T^* = \frac{-\hat{\alpha}\mu\sqrt{4\hat{\alpha}\beta\mu + \hat{\alpha}^2\mu^2}}{\beta\mu} \tag{4}$$

Figure 5 shows the behavior of $\frac{E[n(T)]}{\hat{\alpha}+T\beta}$ varying T for a uniform distribution with parameter $\mu = 1/378s^{-1}$.

Exponential distribution

In the case of exponentially distributed contact duration:

$$F_D(t) = 1 - e^{-\mu t}$$

In this case the optimum point can be expressed in terms of the Lambert function $W(z)$ [5]:

$$T^* = -\beta - \hat{\alpha}\mu - \beta W_{-1}\left(-e^{-\frac{\beta+\hat{\alpha}\mu}{\beta}}\right) \tag{5}$$

Figure 5 shows the behavior of $\frac{E[n(T)]}{\hat{\alpha}+T\beta}$ varying T for an exponential distribution with parameter $\mu = 1/378$.

Pareto distribution

If the contact times are distributed according to a Pareto distribution (power law) with scale k and shape γ :

$$F_D(t) = \begin{cases} 0, & t \leq k \\ 1 - (k/t)^{\gamma}, & t \geq k \end{cases}$$

the optimum value of T can be found numerically by solving the equation:

$$\frac{\lambda T^{-\gamma}\left(-kT^{\gamma}\beta\gamma + k^{\gamma}\left(\hat{\alpha}(\gamma-1) + t\beta\gamma\right)\right)}{\left(\hat{\alpha}+T\beta\right)^2(1-\gamma)} = 0 \tag{6}$$

Real human mobility traces

We aim to calculate the optimum cycle for some real cases using mobility traces available in literature.

Unfortunately, most traces have a granularity that is too coarse for our purposes: for instance, [5] has an inter scanning period of 5 minutes, while in [21] it is 2 minutes.

This problem has been tackled in [7] and overcome by adopting different methods to record the encounters, such as using IEEE 802.15.4 instead of Bluetooth scanning.

Nevertheless, it is worth noting that in all these traces the cumulative distribution of contact duration times approximately follows a power law (Pareto) distribution whose parameters hardly depend on the scenario we are considering. The appropriate value that the authors found in the Haggle IMote trace [21] is γ = 1.5, in NUS data [22] γ = 0.84, while for USC data [7] we have γ = 0.6.

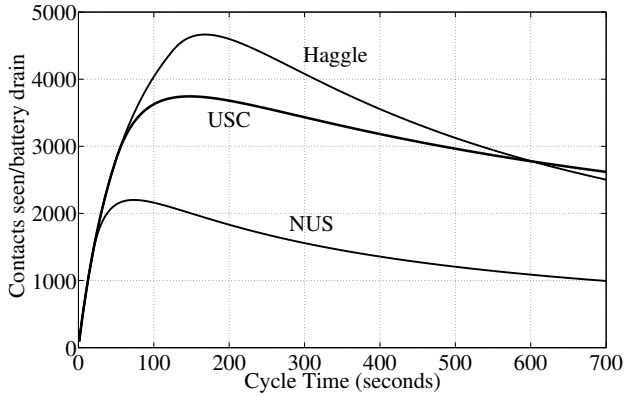


Fig. 6. Contacts detected during battery drain as a function of the cycle, varying the mobility model and according to three different real mobility traces

5. Please note that the Lambert function is the solution of the transcendental equation $z = we^w$. In particular, given that $t > 0$, we need to find the solution in the negative branch $W_{-1}(z)$ that returns a negative value less than -1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

Figure 6 shows the contact ratio detected over the battery drain period varying time T for the distribution inferred from three mobility traces: Haggle Imode [21] $\gamma = 1.5k = 126$, NUS data [22] $\gamma = 0.84k = 18$, and USC data [7] $\gamma = 0.6k = 49.8$.

As we can see, the optimum point largely depends on the considered scenario and typically changes over time. For this reason, in the next section we devise a distributed algorithm that automatically tracks the optimum duty cycle to set on each node.

4 ADAPTIVE DUTY CYCLE

Up until now, we have presented the existence of an optimum cycle where, if all the nodes use that duty cycle, they maximize the number of contacts discovered in a given environment. In this section, we address the problem of how the devices can discover or estimate this optimum cycle.

A straightforward solution could be to use *dedicated infrastructure nodes* (probes) for neighbor discovery, to measure the mobility characteristics of the environments and provide the smartphones with the optimum cycle. But requiring the presence of an infrastructure node in each environment is not easily satisfied.

A more practical solution would be to provide each node with an *adaptive distributed algorithm* to track the optimum cycle value since it varies dramatically over time according to user movement. Consider, for instance, an application that needs to discover new contacts and that runs during the day, when the user is in the subway or in the office, and during the night, when there are no new contacts for hours.

Moreover, we have to note that if all nodes use the same cycle they discover each other, but in a real scenario we need to deal with the possibility of nodes using different cycles. Therefore, when a node wakes up, it can see a subset of the available contacts because some of its neighbor nodes can be asleep.

4.1 Exponential cycles

We developed a solution with a finite set of possible cycles $T_1 ... T_n$ which the nodes can choose from. For instance, the cycles could be $15, 30, 60, 120, 240, 480$ seconds. Using exponential cycles (i.e. $T_i = 2T_{i-1}$) we realize a hierarchy in such a way that when a node with cycle $T = T_i$ wakes up, it detects *all* the nodes in its visibility range with $T_k \leq T_i$ and *some* nodes with $T_k > T_i$. In particular, when it wakes up, one out of two times it can discover nodes with T_{i+1}, one out of four times it can discover nodes with T_{i+2} and so on.

Nodes could easily discover the time they should wake up using their global loosely synchronized clock, as previously explained. For example, a node with $T = 60s$ can wake up when $mod(currTime, 60) = 0$ where *currTime* are the seconds elapsed from Epoch Time (1/1/1970), a typical representation of current time on several operating systems (e.g. on the Android platform, the function *System.currentTimeMillis()* returns the ms from Epoch Time).

This solution was also adopted in [12] and proved to be optimum in terms of discovery latency for discovering new nodes with heterogeneous duty cycles.

4.2 CATNAP Adaptive distributed algorithm

Here we present a very simple 3-rules distributed algorithm that runs on each node with the goal of *detecting* and *tracking* the optimum duty cycle, since the environment surrounding a node could change. We call this algorithm CATNAP.

4.2.1 CATNAP Rules

With reference to a node with an initial cycle T_j, whenever a node wakes up, the proposed algorithm performs these operations:

1 estimates a score $\frac{E[n(T_x)]}{\hat{\alpha}+T_x\beta}$ for all $T_x \geq T_j$.
2 finds the cycle T_k corresponding to the maximum score
3 sets the node cycle as T_{k-1}

where $E[n(T_x)]$ is the average number of nodes discovered when the node wakes up and $mod(currentTime, T_x) = 0$. Thanks to the exponential cycles, a node with cycle T_j can correctly estimate all the scores for all the cycles $T_x \geq T_j$.

4.2.2 Numerical example

Let us clarify with an example. We consider a node with an initial cycle $T = 60s$. This means that this node wakes up every 60 seconds, when $mod(currentTime, 60) = 0$.

The node records the average number of new nodes discovered every time it wakes up, namely $E[n(60)]$, and then calculates a score for $T = 60$ as $score(T = 60) = E[n(60)]/(\alpha + 60\beta)$. This is the contact ratio seen over battery drain as shown in figure 6

Once every two times it wakes up, if $mod(currentTime, 120) = 0$, the nodes could "simulate" the average number of nodes it would have discovered if its cycle were $T = 120$. It can thus calculate $score(T = 120)$ and all the other scores for any $T_x > 60$. However, a node can not simulate what happens for $T_x < 60$ because it is sleeping, so it sets the score for these cycles to 0.

According to the three rules of the CATNAP algorithm, the node sets its cycle to the cycle immediately before the one with the optimum score. For instance, if $score(T = 15) = 0$, $score(T = 30) = 0$, $score(T = 60) = 2.5 \times 10^{-2}$, $score(T = 120) = 3.58 \times 10^{-2}$, $score(T = 240) = 1.42 \times 10^{-1}$ and $score(T = 480) = 9.71 \times 10^{-2}$, we have that $T = 240$ is the cycle that maximizes the ratio of newly discovered peers per amount of energy consumed. Consequently, according to the algorithm, the node sets its cycle to $T = 120$.

4.2.3 Convergence and stability

In a static environment all nodes converge to the same cycle and tend to stabilize at the sub-optimal cycle, as will be showed numerically in section 5 and discussed in what follows.

Let us assume that N nodes have correctly estimated the optimum cycle to the value T_j and correspondingly set their own operating cycles to the value T_{j-1}.

A new node entering into the system can measure the scores for all the cycles $T_x \geq T_{j-1}$ while it estimates $E[n(T_x)] = 0$ for all $T_x < T_{j-1}$ because it finds all the other nodes sleeping

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

during those intervals. However, the new node can correctly calculate the optimum point since it is greater than T_{j-1} and correctly sets its operating cycles to T_{j-1}.

In other words, *the sleeping time of the nodes does not affect the estimation of the scores in the interesting duty cycles* (in which the optimum can be identified) hence the nodes' perceived view of the environment statistics.

We also note that if the new node enters the system with a cycle T_x that is greater than T_j, it will see the greater score for T_x and set its cycle to T_{x-1} according to rule 3 of the algorithm. Iterating, the node decrements its cycle until it selects T_{j-1} that allows the optimum estimation. On the other hand, if the new node enters with $T_x < T_{j-1}$ it can immediately detect the optimum and set its cycle accordingly.

Finally, given that each node continuously monitors the optimum in the system, it can follow the changes in the environment statics that modify the optimum.

We point out that the presented algorithm does not only converge under theoretical hypotheses, but this propriety holds also in the case of errors and fluctuations of the estimations/scores. Let us consider for instance the most challenging scenario in which *all* the nodes in the network start by wrongly considering T_x as the optimum cycle, while the real optimum cycle is $T_{\hat{x}}$. From rule 1, at each time slot, they calculate the score for cycles $T_{x-1}...T_n$. Then if $\hat{x}$ is in the range $[x-1, n]$ they can immediately discover the optimum and change their cycle accordingly by moving to $T_{\hat{x}-1}$; otherwise they jump back to T_{x-2} according to rule 3 and start this reasoning again. This scenario is very similar to the one simulated in figure 8.

4.2.4 Discussion

The basic ideas behind the algorithms are:

- we do not want to alternate environment estimation and a subsequent exploitation but design an algorithm that performs *exploration and exploitation* at the same time
- we do not need to estimate the environmental parameters (such as the distribution of contact duration) and then analytically calculate the optimum cycle, but directly track the optimum point. This can be done thanks to the theory that tells us that there are no local maximum points that do not correspond to the global maximum point.
- we want the algorithm to be distributed on nodes and to be auto-adaptive to the case of a non-stationary environment[6]
- we deliberately want to *keep it simple*, optionally sacrificing optimality
- the node chooses the cycle that is immediately below the optimum to solve the problem of investigating the performance of a shorter cycle. Theory shows us that scores distribute as a concave function so we decided to keep the optimum point *in the visibility scope*. This way we can react to environment changes that move the optimum point towards either shorter or longer cycles.

As a final remark, if the optimum value does not correspond to one of the available duty cycles, the algorithm approaches the optimum by choosing the cycle with the best score among all the available cycles, resulting in a sub-sampling of the contacts seen over battery drain process.

6. This can be easily achieved by using a moving average technique to estimate nodes $E[n(T)]$, such as an exponential moving average.

5 Performance Assessment

We developed a custom event-driven simulator that allowed us to tune the simulation parameters (hence the statistic laws that regulate node mobility) so that we can theoretically calculate the optimum value of the duty cycle for the simulated environment.

The *environment* is the place where nodes **can** meet each other, but an encounter between nodes does not necessarily happen. For example, an environment could represent a campus area or the subway.
Moreover, even if there is an encounter, nodes might not *detect* it because of the duty cycle.
An environment is characterized by some statistics: new nodes arrive with rate λ_e (Poisson) and stay for a certain time (constant or exponentially distributed) with mean μ_e.
Inside the environment visibility is incomplete, but nodes meet with rate λ_u (Poisson) and their encounters last D seconds where D is an RV distributed according to a generic cdf $F_D(t)$. Nodes do not have any a-priori knowledge of these parameters. The rationale behind this model is that we need a dynamic environment where nodes enter and exit the system because we are interested in discovering *new* contacts. At the same time, because of the learning process of the proposed algorithm, a contact between two nodes modifies their state (in particular their score, as described in 4) and could modify their cycle which in turn affects how these nodes could meet others. To see what happens with this "chain reaction", nodes should stay in the environment for a given time and keep meeting other nodes and mutually modifying their state. More details on the simulation process can be found in [23].

5.1 Adaptive algorithm performance assessment

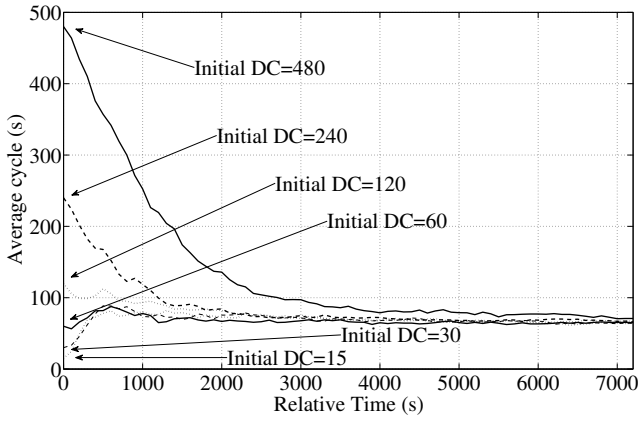
We conducted a simulation campaign with the following parameters: $\mu_e = 1/7200$ (const), $\lambda_e = 1/120$ (exp neg). In this scenario the optimum cycle is $T = 120$. Nodes enter into the system with a random cycle taken from the list of available cycles: 15, 30, 60, 120, 240 and 480 seconds. They then try to auto adapt their duty cycle according to the proposed adaptive algorithm. We plot the average cycle chosen by nodes, grouping nodes by the same initial cycle.

In figure 7 we show how nodes change their cycle during the relative time, i.e. after t seconds they enter the system, in different cases: i) figure 7(a) shows exponentially distributed inter-contact time and contact duration ($\lambda_u = 1/140$, $\mu_u = 1/130$); ii) figure 7(b) shows exponentially distributed inter-contact time and Pareto distributed contact duration ($\lambda_u = 1/140$, $\gamma = 1.2$ and $k = 72$), and; iii) figure 7(c) shows Pareto distributed inter-contact time ($\gamma = 1.06$, $k = 8$) and contact duration ($\gamma = 1.2$, $k = 72$).

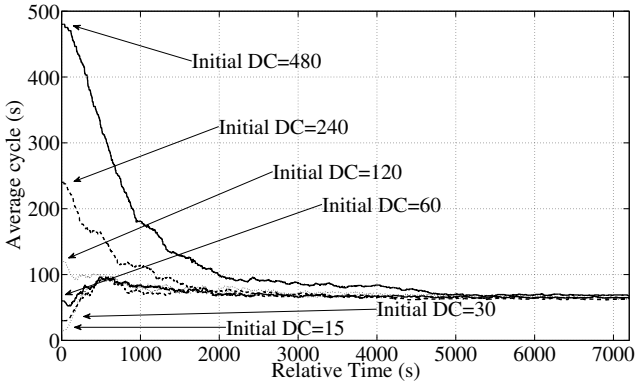
Without regard to the initial cycle, all nodes converge to the cycle $T = 60s$ that is immediately below the optimum $T = 120s$. However, nodes with initial cycles equal to $T = 480$ converge more slowly because they sample the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

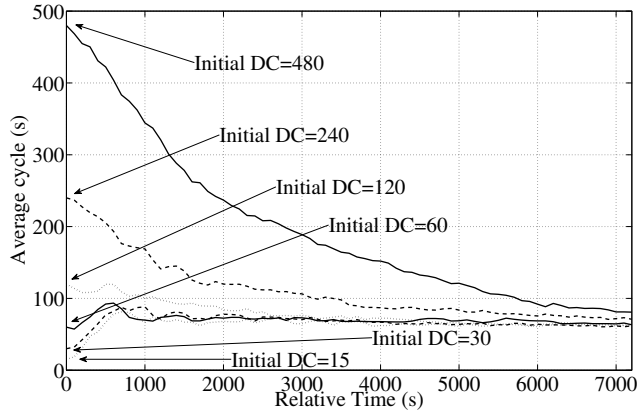
system every 480 seconds. Figure 7(c) shows a slower converge time because of the heavy tail of the Pareto distribution, however this case tests the resiliency of the algorithm under a non-markovian arrival process for which it is not expressly designed to work with.



(a) Exponential inter-contact time/Exponential contact duration



(b) Exponential inter-contact time/Pareto contact duration



(c) Pareto inter-contact time/Pareto contact duration

Fig. 7. Average cycle of nodes for different distributions of inter-contact time and contact duration. According to the proposed algorithm, all nodes converge to the cycle $T = 60s$ that is immediately below the optimum $T = 120s$

We tested the proposed algorithm in a non-stationary environment where we varied the optimum cycle value over time. In particular, we ran a simulation in the same scenario described for figure 7(b), with contact durations following a

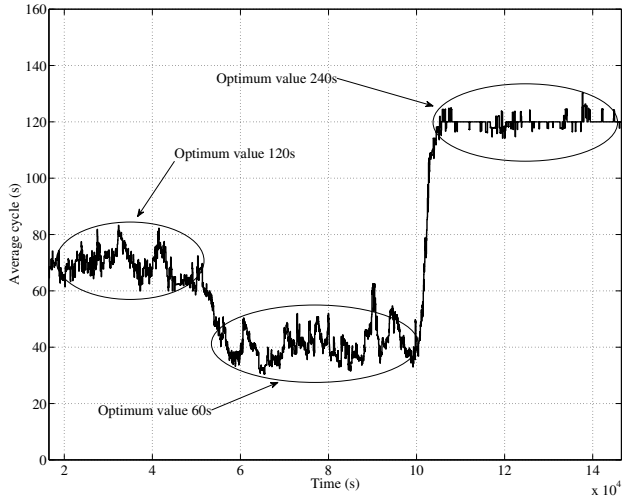


Fig. 8. Average cycle of all nodes in a non-stationary system. Nodes track the optimum cycles that move from T=120, T=60 and T=240 by choosing the immediate sub-optimal cycle values: T=60, T=30 and T=120

Pareto law with parameters that varied during the simulation:

- $\gamma = 1.2$, $k = 72$ for the first third of the simulation. With this distribution the optimum cycle is T=120s
- $\gamma = 2.6$, $k = 42$ for the second third of the simulation. With this distribution the optimum cycle is T=60s
- $\gamma = 2.0$, $k = 206$ for the last third of the simulation. With this distribution the optimum cycle is T=240s

Given that all nodes enter the system with a random cycle and stay for two hours, we plot the average cycle of nodes in the system for nodes that stay in the system for at least one hour, to give the algorithm time to converge. The result of this study is depicted in figure 8. As we can see, the algorithm succeeds in tracking the optimum values by following the sub-optimal values T=60, T=30 and T=120 even if the nodes have no a-priori information about the environment and the environment characteristics change over time. In particular, in this experiment we show the ability of the algorithm to track changes in the environment and consequently adapt the chosen duty cycle, either upward or downward.

5.2 Comparison with the State of the Art

In literature, performance of neighbor discovery protocols is evaluated according to a set of performance metrics. The most used are discovery latency or missing probability. To make a fair comparison, different discovery algorithms are compared using the same energy budget (or alternatively, the same duty cycle) while the choice of how much energy to use is usually not addressed specifically. Instead, in this work we tackle this issue, to determine the optimum amount of energy that maximizes the number of contacts discovered.

The algorithm most similar to ours is [12] where the authors present a Recursive Binary Time Partitioning algorithm (RBTP). Both are synchronous neighbor discovery protocols

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

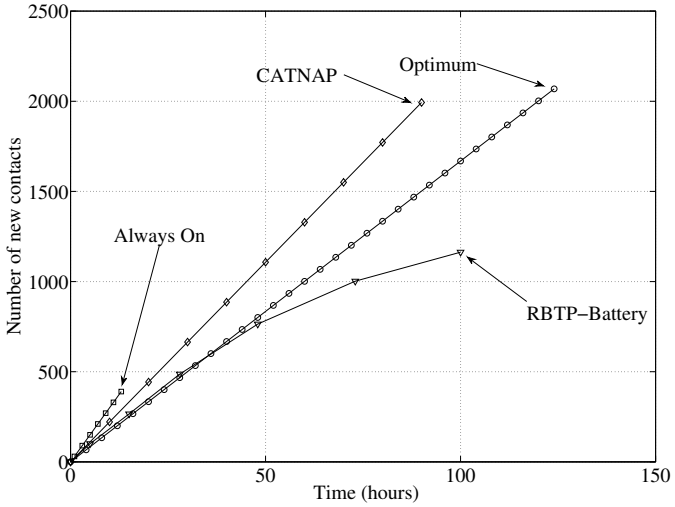


Fig. 9. Number of contacts discovered using a battery charge: ALWAYS-ON vs OPTIMUM vs CATNAP vs RBTP-BATTERY

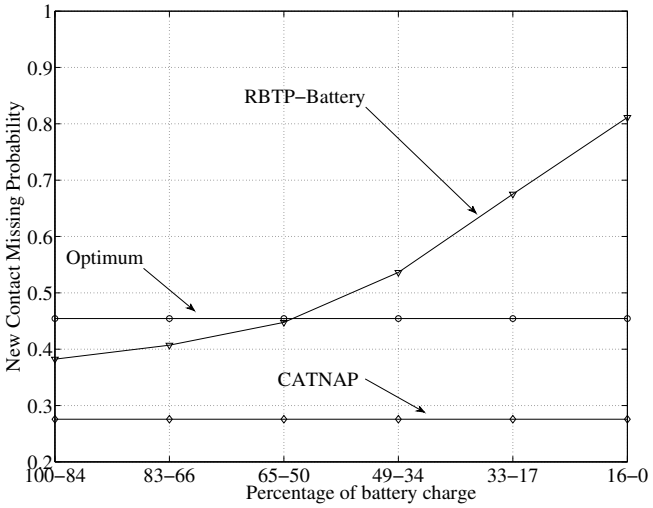


Fig. 10. Probability of missing a contact opportunity varying the battery charge: OPTIMUM vs CATNAP vs RBTP-BATTERY

[7]. In a certain sense this work can be viewed as an RBTP protocol where the duty cycle is chosen dynamically according to the environment.

However, as the authors of [12] state, RBTP allows devices to adapt their number of wake-up instances independently based on their respective energy limitations but in [12] they give no clue as to how a device should choose the number of "wake-up instances" i.e. the energy budget to dedicate to discovery purposes. For this reason, we compare our work with a version of RBTP where the duty cycles change according to the battery level: devices with a high battery charge have a higher duty cycle, while devices running low on battery power increase the sleep time. We call this schema RBTP-BATTERY. This seems a reasonable choice given that several mobile applications limit their functionality according to battery level.

Besides the RBTP-BATTERY algorithm, we compare CATNAP with two other schemes to provide baseline references: OPTIMUM, in which all the nodes use the optimum duty cycle, and ALWAYS-ON, corresponding to nodes that continuously try to discover each other without any duty cycling scheme.

Figure 9 shows the number of discovered contacts during the whole lifetime of a device. The environment is characterized by nodes that enter with an exponential inter-arrival time with an 80s mean, stay in the system for exactly one hour, and have a charge level distributed uniformly between 1% and 100%. Inside the environment, each node has new encounters according to an exponential distribution with parameter $\lambda_u = 1/120$, and the duration of each encounter is distributed according to a power law with parameters (shape: 1.2, index: 72).

The power consumption during the asleep and awake periods are respectively $\alpha = 2.17 \times 10^{-3}$ and $\beta = 1.39 \times 10^{-4}$ (percentage/second). T_{on} is equal to 5s.

7. As shown in [12], even very simple synchronous protocols such as the PRS protocol presented in that paper, outperform asynchronous protocols (such as [3], [24]) for almost all the relevant performance metrics. For this reason we decided to compare with RBTP

As we can see, an ALWAYS-ON approach is the fastest strategy to discover new contacts. It however reduces the duration of the battery to only 12.8h, leading to 400 discovered contacts when the battery runs out of charge. The OPTIMUM approach discovers nodes more slowly, introducing a high missing rate. It does though end up with 2100 new nodes discovered in the life of the battery (125h). The CATNAP algorithm consumes more battery power than the OPTIMUM solution collecting 2000 new contacts in about 90 hours. The RBTP-BATTERY collects $\sim$1150 new contacts at the end of the average life of a device, around half the nodes collected by the OPTIMUM and the CATNAP algorithms. The reason for this result is quite simple: our approach simply avoids wasting energy if there are long contacts. Moreover, by fixing/adapting the duty cycle to the environment, nodes try to approach the same duty cycle and thus discover each other more frequently. Instead, if nodes adapt their duty cycle to their own characteristics (such as their battery charge status), they present a heterogeneous duty cycle distribution.

For the same reason, if we observe the missing probability in fig. 10 the presented approach outperforms the battery adaptive (RBTP-BATTERY) approach. As we can see, OPTIMUM exhibits a higher missing probability than CATNAP, but compensates with more power saving so that at the end of the battery charge it discovers more contacts.

The value of the latency depicted in figure 11 follows a similar behavior because latency is connected with the duration of duty cycles.

6 Implementing Neighbor Discovery

To what extent is peer to peer neighbor discovery supported and ready to be implemented using off-the-shelf mobile devices?

Passing from theory to practice is not straightforward given the features available on commercial hardware and, most of all, the limited support of several discovery related functions by the most used mobile operating systems.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

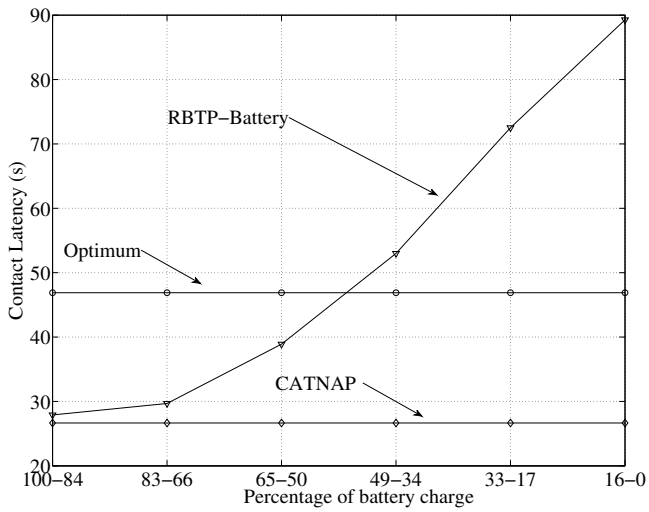


Fig. 11. Contact latency varying the battery charge: OPTIMUM vs CATNAP vs RBTP-BATTERY

In this section we show and discuss the current support, both in terms of hardware and software facilities, for implementing peer-to-peer neighbor discovery on all the main mobile platforms and devices. Finally, we show an implementation of the proposed energy saving neighbor discovery mechanism, to prove its real feasibility and provide technological insights.

6.1 Support of neighbor discovery on mobile platforms

From the hardware perspective, mobile devices and tablets are general-purpose devices, not explicitly designed for extensive use in opportunistic and peer-to-peer communications but rather to connect with remote servers. However, the needs for location based services and the onset of Internet of Everywhere started changing the game. The introduction of Bluetooth Low Energy (BLE) is an example, allowing for instance to discover the presence of a BLE device and consequently open a specific page, thus realizing a location based service. Indoor navigation, home automation and the control of "things" are other services where this type of communication is needed.

6.1.1 Hardware support: network technologies for peer-to-peer neighbor discovery

WiFi ad-hoc mode: Historically, one of the most used technologies allowing peer-to-peer communication is WiFi ad-hoc mode (IEEE 802.11 Independent Base Service Set, IBSS). This technology enables peer communication without any kind of association or asymmetric roles among peers (such as in 802.11 AP-STA communications) – for this reason this technology played a relevant role in the field of mobile ad-hoc networks. Neighbor peers can exchange data at WiFi basic rate (typically 1Mbps) whenever they are close enough. Although radio devices equipped in smartphones theoretically support this mode, the lack of a native operating system support discourages its real usage.

WiFi Direct: More recently, WiFi Direct, a specification from the Wi-Fi Alliance, has begun to be supported by Android and Blackberry mobile systems and many other devices such as laptops or game consoles. Unlike WiFi ad-hoc mode, a Wi-Fi Direct device has to implement both the role of a client and the role of an APP (usually referred to as SoftAP). Communication is provided with security (WPA2) and power management seeing as it is functionally equivalent to traditional Wi-Fi infrastructure-based communication. Neighbor discovery is performed by alternating two states: a search state in which the device sends Probe Requests on several channels, and a listen state in which the device listens for other neighbors' Probe Requests. A node remains in each state for a random time typically between 100 ms and 300 ms, but it is up to the implementation to decide whether to introduce sleeping cycles for energy saving [25].

Bluetooth: Although having a narrower connection range than WiFi, Bluetooth plays a prominent role in mobile neighbor discovery due to its historical availability on mobile devices. Traditionally, implementing neighbor discovery by means of periodical Bluetooth scan operations can be very slow [7] and power consuming. However, with the recent introduction of *Bluetooth Low Energy*, BLE, (Bluetooth Core Spec. V4.0) in several consumer market products, this is changing. To implement neighbor discovery, BLE devices periodically emit advertising information on the three advertising channels and listen to advertising information originating from other devices [26]. While the duration of the scanning is bound to 10.24s by the standard, the time between subsequent scanning operations to achieve energy saving is implementation dependent.

Other technologies: In sensor networks, Wake-On-Radio (WOR) functionality enables the radio to periodically wake up from sleep mode and listen for incoming packets with minimal CPU interaction [8]. This functionality enables very energy efficient neighbor discovery as it allows synchronous operations. However, to the best of our knowledge, these technologies are still not available on mobile devices.

6.1.2 Software support: mobile operating systems and frameworks

We explored native support of the main mobile operating systems for unattended peer-to-peer discovery. Currently, the most used mobile platforms introduce several limitations on what an application can do in the background, mostly for security and power efficiency reasons.

Android: As for radio interface management, since Android 4.0, WiFi Direct is supported but unattended operations are not yet possible, requiring explicit user interactions to discover and associate with new peers. The same issue appears for Bluetooth when used in stadard configuration, whilst in Bluetooth Low Energy and since Android 5.0 mode some workarounds are possibles [9]. Android does not support WiFi Ad-Hoc mode (see Bug82 [10]) but several Android-based

8. http://www.ti.com.cn/cn/lit/an/slaa459a/slaa459a.pdf
9. http://altbeacon.github.io/android-beacon-library/index.html
10. http://code.google.com/p/android/issues/detail?id=82

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

custom roms provide this support. In particular, Cyanogen mod (www.cyanogenmod.org/), one of the most used custom roms, offers a set of APIs to turn on, off and configure the wireless interface in an ad-hoc mode.
To implement the background operations needed once a neighbor peer is discovered, Android supports background code execution by means of the Service structure.

iOS: iOS offers only limited support for background operations. An application can set a "background mode" choosing among a predefined list and be notified by the operating system when some events occur (e.g. a location update). Once notified, the application has at most 10 seconds to execute its background code, while other facilities (e.g. queues) can be used to process the previously acquired information when the user explicitly opens/resumes the application. Currently iOS supports neither WiFi direct nor WiFi ad-hoc mode. In the recent iOS 7, multi-peer connectivity has been introduced, but this functionality is limited to 6 participants and lacks corresponding background support (no background mode is currently available). Since the arrival of Bluetooth 4, cross device communication has become possible also with Android devices or with embedded BLE devices (e.g. iBeacon [1]). However, it is still not possible to execute long and complex background data transmission due to the above mentioned time limitation.

Cross platform frameworks: One of the most used cross platform frameworks is the AllSeen Alliance framework [11]. Initially based on the AllJoyn open source project, this framework wants to provide support for implementing the Internet of Everything, where obviously neighbor discovery plays a fundamental role. The actual discovery mechanism adopted in the framework is transport-dependent: on Wi-Fi, it's a lightweight IP multicast protocol, and on Bluetooth, it's an extended inquiry response (EIR) and suspended discovery protocol (SDP) query.

6.2 Proof of concept

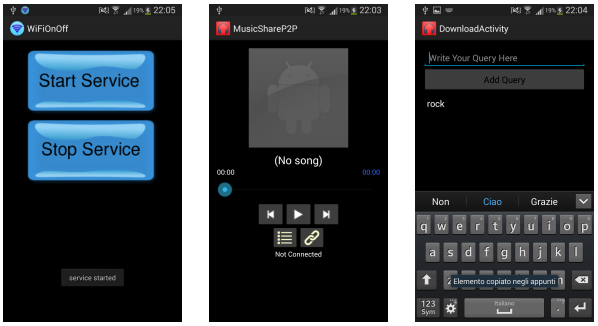
Based on the above-mentioned support, we proved the feasibility of the proposed neighbor discovery solution by implementing a mobile application on Android Platform. We used WiFi Ad-hoc mode provided by the Cyanogen Mod described above.

In particular, we implemented two distinct applications:

- A *background service* (fig. 12(a)) that implements the proposed neighbor discovery algorithm. This service provides the network discovery functionality that can be used by one or more applications. In this way we separate the application logic from the discovery implementation.
- A demo application called *music share* (fig. 12(b), 12(c)) that uses the previous service to be notified when a contact opportunity occurs. The goal of the application is to share music files in a given directory with other phones in the vicinity in an unattended mode, representing an example of an opportunistic application.

The code is available on a public code repository [12].

11. https://allseenalliance.org/
12. https://github.com/netgroup/hts-cycle



(a) Service screenshot (b) Main screen of the music share app (c) Users can choose the music they want to download

Fig. 12. The proof-of-concept service and application

6.2.1 Algorithm implementation and pseudocode

We can sum up the proposed discovery operations as follows:

- *Every node cycles between an active state (ON) and a sleep state (OFF)*. Given a duty cycle T each node stays ON for $T_{on} = 5s$ and OFF for $T - T_{on}$ seconds. A node wakes up when $mod(currTime, T) = 0$ where *currTime* can be obtained by the function *System.currentTimeMillis()*.
- *During the ON period, every node frequently emits discovery packets and listens for those coming from other nodes*.
- *On packet arrival, each node could notify an application that a new contact is available*. The application, in turn, could decide to perform application dependent operations such as initiate a file transfer activity with the newly discovered peer. The occurrence of such operations prevents the nodes from switching to the OFF state for the time needed to complete the data transfer operation.
- *When the ON phase is terminated a node updates its statistics and computes the next duty cycle*, according to the algorithm described in section 4.
- *When the ON phase is terminated and no other communication is in progress, the node goes into the OFF state by turning its Wi-Fi radio interface OFF for T_{off} seconds.*

We sum up all the operations, together with the proposed adaptive algorithm, using the following pseudo-code (algorithm 1):

6.2.2 Implementation details

Specifically, we implemented the algorithm as an Android Service so that: i) one or more applications could bind to that service; ii) the service is in charge of emitting/receiving presence packets during the ON phase; iii) the service notifies the application when a new contact is discovered; iv) the applications could prevent the service from turning the wireless interface off. Before shutting down the wireless interface and putting the smartphone in sleep mode, the service asks if there are some applications that still need to use the network (e.g. a download is in progress) by sending a Broadcast Intent. Then it sets itself as a Broadcast Receiver to intercept that intent so that if other apps do not block the intent, the service gets its intent back and can safely turn the wireless interface off

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

Algorithm 1 Duty cycle algorithm

```
downloadInProgress ← false
on packet arrival: call application logic
while true do
  Turn on and configure the radio interface
  Listen for presence packets
  for i=1 to i=5 do
    emit presence packet
    wait 1 sec
  end for
  Stop listening for presence packets
  for cycle IN cycles do
    if cycle ≥ MyCycle then
      UpdateScore(cycle)
    end if
  end for
  indexCycleWithMaximumScore ← getMax(scores)
  MyCycle ← cycle[indexCycleWithMaximumScore - 1]
  if not downloadInProgress then
    turn off the radio interface
  end if
  sleepTime ← MyCycle - (getCurrentTime() % MyCycle)
  sleep(sleepTime)
end while
```

without interrupting any operations performed by applications. When we are in the ON state, the service periodically emits presence packets with the name of the registered applications and listens for other nodes' presence packets. When the service receives a packet containing the name of one of its registered applications, it notifies the application of the new contact opportunity through an Intent. The service's wake-up procedure is implemented through the Alarm Manager whose goal is to periodically and automatically acquire the CPU wakelock, call a function that implements our logic, and then release the power lock so that the phone can go back to sleep.

6.2.3 System parameters configuration

We set T_{on} to 5 seconds considering 2 seconds for clock drift, and 2 seconds for switching on the WiFi interface, plus a second of margin. We set the list of the possible cycles to be $15, 30, 60, 120, 240, 480$ seconds, as a good balance for the current battery storage capacity (see fig 2).

During the ON period, the application emits one packet per second. During experimentation we did not see any meaningful energy saving margin by modifying the rate of discovery packet emission: one advertising packet per second seems enough to cope with the problem that terminals are not perfectly synchronized.

Finally, we empirically noticed that, using the Android Alarm Manager rather than other strategies (such as threads and sleeps) provides the best results in terms of power consumption as documented in [11].

6.2.4 Discussion

This implementation is a proof-of-concept of the viability of the proposed solution that however remains *communication technology independent*. Indeed, contrary to other techniques available in literature, the presented algorithm does not use any specific facility of the underlying network technology. It only needs the execution of a background discovery service and the support for sending and receiving broadcast packets and the ability to set timers. Thus, different technologies can be used instead of WiFi ad-hoc. Changing the network technology has an impact on the duration of T_{on} as it also depends on the capacity of turning on and off the network interface, and on α and β that in turn affect the *scores* in the optimization formula. As a final remark, the expected variance of the new meeting process affects the list of possible cycles, although very long cycles offer a rather negligible benefit in terms of energy saving, as devices discharge constantly also when in sleeping mode, as reported in figure 2.

7 Related work

Discovery protocols can be classified according to their degree of coordination, in synchronous and asynchronous.

Asynchronous discovery protocols

The great majority of available works (e.g. [27]) makes no assumption on the presence of a global reference clock among all the nodes and are thus considered totally independent. Instead, we focus on modern smartphones that offer global synchronization by default, as already discussed in section 2, clearly permitting far better neighbor discovery performance [12]. However, several analogies with asynchronous discovery protocols can be drawn as the problem has often been tackled under several different perspectives.

The problem is often formulated in terms of a trade-off between energy consumption and missing probability [28] [8] [29] [3] or the discovery latency [30] [24] [31]. For instance in [8] the authors propose to choose the inter-probe times based on the movement speeds of devices so as to detect more than 99% of encounters in their case scenario.

Instead, in this work our goal is to maximize the number of contacts *in the long run*, hence we are not interested in a high detection probability per se, although we too auto-adapt the duty cycle to the mobility statistics. Similarly, in [30], the authors propose a scheme that auto adapts the duty cycle but their goal is to provide a high delivery ratio and a low delivery latency. Disco [3], U-Connect [24], [32] and the "birthday protocols" proposed by McGlynn and Borbash in [29] adopt a slotted time model and investigate the most energy efficient way to make two or more nodes wake up in the same slot. This is a very challenging goal and their results can be applied to many scenarios, especially wireless sensor networks. However, in our scenario, the presence of global synchronization changes this issue and allows us to provide an easier solution to this problem. In [33], Wang et al. propose STAR, a contact-probing algorithm that adapts to the contact arrival process. It only addresses the energy consumption issue for the data transmission phase while the energy spent for searching for neighbors is usually far from negligible (99.5% in the scenario reported by [34]).

A very recent work [10] proposes an adaptive duty cycle for opportunistic networks based on a cooperative approach

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

and a protocol to schedule the node wake-up intervals. Their approach adaptively schedules wake-up periods of mobile nodes so that a node stays asleep during inter-contact times, when contact probing is unnecessary, and only wakes up when a contact with another node is likely to happen. The authors use the statistics of previous encounters to optimize the re-discovery process between a pair of nodes. They minimize the duty cycle, assuring that the probability of two nodes to communicate is above a given threshold "p" (called performance requirement) which can be set according to the nodes' battery life. On the contrary, *we tackle the problem of the initial meetings* whose statistics depend on the environment.

In [9], the authors introduce an adaptive scheme specifically designed for Bluetooth enabled devices. They adjust the Bluetooth scan settings according to the measured mobility context to decrease the overall power consumption of the discovery process: they reduce the energy consumption by 50%. On the contrary, *our approach is not technology dependent* and allows to save much more energy (e.g. 600% or 1000%) by performing *very slow scans* that consequently could lead to non-negligible missing probability if compared with these works. We can accept a significant missing contact probability, given that *our goal is to allow a node to discover as many new nodes as is permitted by its energy budget and the environmental conditions*.

In [4], the authors present eDiscovery a discovery protocol for energy efficient discovery using Bluetooth technology. In that work, they use recent android smartphones like we do, but they do not exploit the synchronization facility offered by the operating system. Moreover, once again, we have a different goal: theirs is to maximize the ratio of discovered peers.

Finally, since most smartphones have accelerometers, in [35] authors pursue a different strategy and trigger Bluetooth scanning operations according to user movements that in turn result in user contact changes.

Synchronous discovery protocols

To improve the performance of the discovery protocol, very few recent works have been carried out proposing the synchronous discovery protocol, because the requirement of a global clock source is often considered unrealistic for many application scenarios. This consideration is changing with the arrival of smartphones.

RBTP [12], like our solution, is the only other discovery protocol specifically designed for mobile phones. Indeed, they propose a synchronous discovery protocol based on the presence of loose synchronization, easily available by using the NTP protocol. Their experiments show that smartphones can be synchronized within 100ms if the synchronization is performed every 6 hours; this is near our experimental findings [11], even though the delay of wireless activation/deactivation is in the order of seconds. Their solution is based on a duty cycle scheme where the ON periods are distributed according to a binary slotted configuration. Nodes can select their own duty cycle independently, "based on their respective energy limitations", but *how to choose the right duty cycle given the energy budget is not discussed in the paper*. They show that their quasi-synchronous solution outperforms any other asynchronous discovery protocols in terms of missing probability and discovery latency. Their simulations are based on a random walk mobility model. Our proposal differs in a few key ways.

In [36], Campus-mur and Loureiro proposed a synchronous Energy efficient Wi-Fi Discovery technique based on the presence of an infrastructure of access points to provide the nodes with a reference clock. This protocol is designed for cluster and not individual discovery.

In the context of mobile opportunistic networks, to the best of our knowledge, this is the first work that i) formalizes the problem of a maximum discoverable number of peers with an energy budget; ii) provides a theoretical analysis of this problem, and; iii) presents a simple yet effective and easily deployable algorithm to address this problem.

Analytic works and models

In [37] the authors propose a general model to derive the pairwise inter-contact times modified by a duty cycling policy. That work presents the same scenario as ours, but their goal is different: they want to improve message forwarding delays and network lifetime. Finally, there are also many works that study the statistics of contacts between nodes, such as [14], [15], [16] and [17] both from practical (trace-based) and theoretical (stochastic process) points of view.

8 Conclusions

In this paper we presented an energy saving solution to discover new contacts on opportunistic and delay tolerant networks by means of duty cycling. We discovered the presence of a natural optimum cycle that maximizes the number of detected contacts over the whole life of the mobile device. Through an analytic model we provided handy formulas to calculate this optimum value using several mobility models, including some derived from three real mobility traces. We proposed an adaptive distributed 3-rules algorithm that tracks the optimum cycle to use in stationary and non-stationary environments. Finally, we evaluated the performance and the convergence time of the proposed algorithm and compared it with the state of the art. We implemented the proposed strategy on Android devices to demonstrate its feasibility.

References

[1] A. Inc. Ibeacon. [Online]. Available: https://developer.apple.com/ibeacon/

[2] W. Wang, V. Srinivasan, and M. Motani, "Adaptive contact probing mechanisms for delay tolerant applications," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, ser. MobiCom '07. New York, NY, USA: ACM, 2007, pp. 230–241.

[3] 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*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 71–84.

[4] B. Han and A. Srinivasan, "ediscovery: Energy efficient device discovery for mobile opportunistic communications," in *Proceedings of the 2012 20th IEEE International Conference on Network Protocols (ICNP)*, ser. ICNP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–10.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2471299, IEEE Transactions on Mobile Computing

[5] N. Eagle and A. (Sandy) Pentland, "Reality mining: sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, Mar. 2006.

[6] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2009-05-29)," May 2009.

[7] Y. Wang, B. Krishnamachari, and T. Valente, "Findings from an empirical study of fine-grained human social contacts," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, 2009, pp. 153–160.

[8] M. Orlinski and N. Filer, "Movement speed based inter-probe times for neighbour discovery in mobile ad-hoc networks," in *Ad Hoc Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Zheng, N. Mitton, J. Li, and P. Lorenz, Eds. Springer Berlin Heidelberg, 2013, vol. 111, pp. 316–331.

[9] C. Drula, C. Amza, F. Rousseau, and A. Duda, "Adaptive energy conserving algorithms for neighbor discovery in opportunistic bluetooth networks," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 1, pp. 96–107, Jan 2007.

[10] W. Gao and Q. Li, "Wakeup scheduling for energy-efficient communication in opportunistic mobile networks," in *Proceedings of the 32th IEEE Conference on Computer Communications*, ser. INFOCOM, 2013.

[11] L. Bracciale, P. Loreti, and G. Bianchi, "Human time-scale duty cycle for opportunistic wifi based mobile networks," in *Proceedings of the Tyrrhenian International Workshop on Digital Communications*, 2013.

[12] D. Li and P. Sinha, "Rbtp: Low-power mobile discovery protocol through recursive binary time partitioning," *IEEE Transactions on Mobile Computing*, vol. 13, no. 2, pp. 263–273, 2014.

[13] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra, "Per-frame energy consumption in 802.11 devices and its implication on modeling and design," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[14] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *Proceedings of MobiHoc '09*. New York, NY, USA: ACM, 2009, pp. 299–308.

[15] A. Passarella and M. Conti, "Characterising aggregate inter-contact times in heterogeneous opportunistic networks," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part II*, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011.

[16] V. Conan, J. Leguay, and T. Friedman, "Characterizing pairwise inter-contact patterns in delay tolerant networks," in *Proceedings of AUTONOMICS07*, ICST, Brussels, Belgium, Belgium, 2007, pp. 19:1–19:9.

[17] H. Cai and D. Y. Eun, "Crossing over the bounded domain: From exponential to power-law intermeeting time in mobile ad hoc networks," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 5, pp. 1578–1591, Oct 2009.

[18] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma, "Explaining the power-law distribution of human mobility through transportation modality decomposition," *arXiv preprint arXiv:1408.4910*, 2014.

[19] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[20] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnovic, "Power law and exponential decay of intercontact times between mobile devices," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 10, pp. 1377–1390, 2010.

[21] P. H. Augustin Chaintreau, "Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding," 2006 Computer Laboratory, University of Cambridge, Tech. Rep., Feb. 2005.

[22] A. Natarajan, M. Motani, and V. Srinivasan, "Understanding urban interactions from bluetooth phone contact traces," in *Proceedings of the 8th international conference on Passive and active network measurement*, ser. PAM'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 115–124.

[23] L. Bracciale, P. Loreti, and G. Bianchi, "Simulating the statistics of the first meetings using dynamic open environments," in *4th IEEE Track on Collaborative Modeling and Simulation (CoMetS'14)*, 2014.

[24] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar, "U-connect: A low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proceedings of IPSN '10*. New York, NY, USA: ACM, 2010, pp. 350–361.

[25] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with wi-fi direct: overview and experimentation," *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 96–104, June 2013.

[26] J. Liu and C. Chen, "Energy analysis of neighbor discovery in bluetooth low energy networks," *Nokia.(nd)*, 2012.

[27] M. Bakht, M. Trower, and R. H. Kravets, "Searchlight: won't you be my neighbor?" in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 185–196.

[28] W. Feng and S. Li, "Energy efficient terminal-discovering in mobile delay tolerant ad-hoc networks," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*, Oct 2013, pp. 465–470.

[29] 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 &Amp; Computing*, ser. MobiHoc '01. New York, NY, USA: ACM, 2001, pp. 137–145.

[30] Y. Xi, M. Chuah, and K. Chang, "Performance evaluation of a power management scheme for disruption tolerant network," *Mob. Netw. Appl.*, vol. 12, no. 5, pp. 370–380, Dec. 2007.

[31] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proceedings of MobiSys '10*. New York, NY, USA: ACM, 2010, pp. 255–270.

[32] B. J. Choi and X. Shen, "Adaptive asynchronous sleep scheduling protocols for delay tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 9, pp. 1283–1296, 2011.

[33] W. Wang, M. Motani, and V. Srinivasan, "Opportunistic energy-efficient contact probing in delay-tolerant applications," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1592–1605, Oct. 2009.

[34] N. Banerjee, M. Corner, and B. Levine, "An energy-efficient architecture for dtn throwboxes," in *INFOCOM 2007*, May 2007, pp. 776–784.

[35] W. Hu, G. Cao, S. Krishanamurthy, and P. Mohapatra, in *ICDCS*, July 2013, pp. 155–164.

[36] D. Camps-Mur and P. Loureiro, "E2d wi-fi: A mechanism to achieve energy efficient discovery in wi-fi," 2014.

[37] E. Biondi, C. Boldrini, A. Passarella, and M. Conti, "Duty cycling in opportunistic networks: intercontact times and energy-delay tradeoff," CNR, Tech. Rep., December 2013.

Lorenzo Bracciale has been an Assistant Professor in the University of Rome Tor Vergata since June 2013. He obtained his Ph.D. with a thesis on peer-to-peer multimedia communications. He collaborated with several companies and other researchers worldwide for projects regarding delay tolerant networking and mobile applications. His current research interests include autonomous and self-organizing systems, either in Wireless Sensor Networks or in Mobile Networks.

Pierpaolo Loreti received his Degree in Electronic Engineering (cum laude) in July 1998 and his Ph.D. in telecommunications and microelectronics in June 2002, from the University of Rome Tor Vergata. From 2002 to 2005 he was a Researcher at the National Telecommunications Inter-University Consortium (CNIT). Since 2006 he has been a Researcher of the Dep. of Electronic Engineering and an Adjunct Professor at the Internet Engineering Course at the University of Rome Tor Vergata. Since 1998 he has worked on several European and national projects performing research and coordination activities. His research activity spans different topics in the areas of wireless and mobile networks, framework design, analytic modeling, performance evaluation through simulation and test-bedding.

Giuseppe Bianchi has been Full Professor of Networking at the University of Rome Tor Vergata since January 2007. His research activity includes Wireless LANs, IP networking, privacy and security, traffic monitoring, and is documented in about 200 peer-reviewed international journals and conference papers. He has had (and has) coordination roles in many large scale European research projects in the FP6, FP7 and in the upcoming H2020 programs. He has served as associate editor for IEEE/ACM Transactions on Networking, area editor for IEEE Transactions on Wireless Communication, and area editor for Elsevier Computer Communication. He has chaired several international networking conferences and workshops, including IEEE Infocom 2014, ACM SRIF 2013, ACM Wintech 2011, IEEE WoWMoM 2010, etc.