WLAN Interface Management on Mobile Devices

by

Mohammad Hossein Falaki

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

The number of smartphones in use is overwhelmingly increasing every year. These devices rely on connectivity to the Internet for the majority of their applications. The ever-increasing number of deployed 802.11 wireless access points and the relatively high cost of other data services make the case for opportunistic communication using free WiFi hot-spots. However, this requires effective management of the WLAN interface, because by design the energy cost of WLAN scanning and interface idle operation is high and energy is a primary resource on mobile devices.

This thesis studies the WLAN interface management problem on mobile devices. First, I consider the hypothetical scenario where future knowledge of wireless connectivity opportunities is available, and present a dynamic programming algorithm that finds the optimal schedule for the interface. In the absence of future knowledge, I propose several heuristic strategies for interface management, and use real-world user traces to evaluate and compare their performance against the optimal algorithm. Trace-based simulations show that simple static scanning with a suitable interval value is very effective for delay-tolerant, background applications. I attribute the good performance of static scanning to the power-law distribution of the length of the WiFi opportunities of mobile users, and provide guidelines for choosing the scanning interval based on the statistical properties of the traces. I improve the performance of static scanning, by 46% on average, using a local cache of previous scan results that takes advantage of the location hints provided by the set of visible GSM cell towers.

Acknowledgements

I would like to thank my supervisor, Prof. S. Keshav, for his advice and mentorship. He helped me identify this problem and provided guidelines in every stage of the research. Without his help this work would have not been successful.

Earl Oliver helped me with the user interaction experiments. He developed the logging tool for BlackBerry. My thanks to him for both his help and his enthusiasm.

I enjoyed working in the Tetherless Computing Lab with a group of motivated, hard working and fine researchers: Earl Oliver, Shimin Guo, Nabeel Ahmed, Aaditeshwar Seth, David Hadaller, Sumair Ur Rahman, Usman Ismail, and Omer Beg. I learned a lot from each of them.

I would like to thank my thesis readers, Prof. Bill Cowan, and Prof. Johnny Wong, for their comments and feedback that helped me improve the thesis.

My English mentors, Nicole Keshav and Sholeh Shahrokhi, taught me how to intelligently present my ideas in words. I will always benefit from their great job in teaching, and will always appreciate it.

I enjoyed discussing my ideas with a great friend, Majid Mirbagheri, who helped me with his critical comments.

Finally I should thank all those who volunteered to participate in this study. I cannot mention their names but I will always remember their help.

This research was supported by grants from the National Science and Engineering Council of Canada, the Canada Research Chair Program, Cisco Research, and Nokia Research.

Dedication

I would like to dedicate this work to my parents:

Mehri Ahardehi and Hassan Falaki

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The number of smartphones in use is increasing every year, and sales of smartphones will overtake laptop sales in 2009 [1]. These mobile devices are equipped with multiple Network Interface Connectors (NIC) including a WLAN interface for 802.11, and a cellular (GSM or CDMA) interface. Among the available data communication technologies on current smartphones, 802.11 has some unique features that make it the best choice for data communication. First, it is a relatively short-range radio technology, therefore it offers higher data rates and consumes less energy per byte [1] as seen in Figure 1.1. Also, using freely available organizational or home WiFi networks is preferable to communication over expensive cellular data plans.

To find WLAN connectivity opportunities a smartphone should scan the environment regularly. Once a usable access point [25] is discovered the interface can associate with it and start communication. Therefore, if a device has pending data to send or receive, the WLAN interface should be active and scanning even when it is not associated with any access points. However, because of the carrier sense nature of the 802.11 standard, WLAN interfaces consume considerable energy for scanning [2] and even when idle [1].

This Thesis addresses the WLAN interface management problem, as formally defined in Section 1.2. Section 1.3 expresses the objectives of the thesis, and Section 1.4 enumerates the main contributions of my research. Finally Section 1.5 outlines the organization of the rest of the thesis.

1.1 WLAN Scanning

WLAN scanning can be either passive or active. When performing a passive scan, the WLAN interface iteratively listens on all 802.11 channels for periodic access point beacons. In active scanning, the station interface sends a probe request frame on each channel. This wireless frame will initiate a probe response from each

[1]http://news.bbc.co.uk/2/hi/technology/7250465.stm

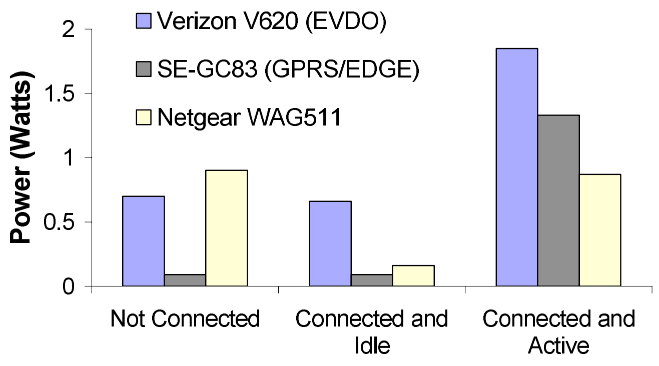

Figure 1.1: Power consumption of three wireless data communication technologies in different states of the interfaces, from the study by Agarwal *et al.* [1]. Verizon V620 is an EVDO interface, SE-GC83 is and EDGE interface, and Netgear WAG511 is a Wi-Fi interface.

access point that is operating in that channel and receives that frame. Each beacon or probe response frame contains essential information required by the station computer to associate to the access point.

Active scanning consume more energy, and in return, can discover access points that would not broadcast their presence. However, both active and passive scanning are energy consuming. The work in this thesis does not make any assumption about the type of the WLAN scan.

1.2 Problem Statement

Consider an always-on battery-operated, hand-held mobile device, equipped with an 802.11 interface. Assume that the device always has data to send. Both background and interactive processes on such a device need network connectivity using the WLAN interface in the least intrusive way [36] to the user. Considering the power consumption profile of 802.11 interfaces, *what is the best strategy to scan for WLAN wireless access opportunities?*

1.3 Objectives

This thesis tries to identify effective scanning strategies that work best for smartphones and other hand-held mobile devices (such as PDAs). I use traces from real-world measurements to evaluate the performance of the proposed scanning strategies.

I use two performance metrics to evaluate the scanning strategies: number of scans and missed opportunity. Missed opportunity is the ratio of the wireless time that the scanning strategy detects to the total available time.

The goal is to find a scanning strategy that is pareto optimal, when evaluated using real-world traces, and also implementable on current hardware and software systems.

1.4 Contributions

This thesis makes the following contributions:

1. Discovering that most mobile users are covered by WiFi access points for most of the day
2. I identify a power-law distribution in the length of the WiFi connectivity opportunities of mobile users
3. I exploit the long tail distribution of connectivity opportunities with a simple and practical scanning strategy
4. I improve the performance of any scanning strategy with a cache of previous scan results using context hints
5. I study the effect of user-initiated scans on the proposed scanning strategy

1.5 Thesis Organization

Chapter 2 discusses related research results and how the results of this thesis can complement already existing systems. Chapter 3 presents a formal modeling of the problem and a hypothetical optimal algorithm that assumes knowledge of the future connectivity opportunities. In the absence of such information I propose four heuristic scanning strategies in Chapter 4 and analyze their expected performance. In Chapter 5 the measurements methodology and results are presented, and in Chapter 6, I outline the design and implementation of the simulation tool that I used to evaluate the performance of the proposed strategies. Chapter 7 presents the results of the simulations based on the trace-driven simulation . Based on the conclusions of this chapter a general WLAN interface management strategy is proposed in Chapter 8, Finally Chapter 9 concludes the thesis and outlines possible future work.

Chapter 2

Related Work

This chapter presents the work related to WLAN interface management. In each case I discuss how the results of this thesis can complement the previous research results and vice versa.

2.1 Delay Tolerant Networking

Delay Tolerant Networking (DTN) [10] is gaining popularity as a practical paradigm for mobile application developers. Most delay-tolerant mobile applications opportunistically exchange information between mobile devices when users encounter and do not take advantage of any infrastructure. Examples of such applications are mobile social networking [28], and mobile file-sharing applications. Another class of delay-tolerant applications on mobile phones exploit opportunities for connectivity to the Internet using wireless hot spots. Examples of such applications include delay-tolerant database applications [23], content update systems [2], sensor data collecting applications [9, 7], or even the classical email application.

Both classes of applications, when using WiFi, need to scan the environment for opportunities. However they have different scanning requirements. When looking for ad-hoc contacts, scanning should continue even after finding a contact. Wei *et al*, [34] study the trade-off between probability of missing a contact and the frequency of scanning (they refer to scanning as "probing") in this context. They use Bluetooth measurements to evaluate the performance of their proposed algorithm.

When trying to opportunistically use infrastructure, as soon as a usable access point is discovered, scanning can stop. Therefore the mechanisms and strategies proposed by Wei *et al*, [34] are not usable in this context.

Mechanisms such as caching [31] have been proposed for seamless handoff between 802.11 access points for delay-sensitive applications such as voice over IP.

2.2 Context-aware Computing

The idea of using context information on smartphones to conserve energy [30, 32] is fairly recent. Rahmati and Zhong [30] assume that an application needs to send data at specific points in time. They use context information such as GSM cell tower IDs, and accelerometer readings along with history to decide if it is better to scan for a WiFi access point or directly use the EDGE network. In their work the WLAN interface is always preferred because of its better energy trade-off. To evaluate their proposed algorithms they use traces from a measurement study. Their traces, which are available on CRAWDAD [15], have been used in this thesis as the secondary measurement set.

CABMAN [32] is a system that warns the smartphone user when it detects that they will run out of battery charge before their next "charging opportunity." It uses the currently associated cell tower ID as a landmark hint along with call and charging history to build a model that can make such a prediction. They use the MIT's Reality mining [1] dataset to evaluate the performance of their system.

2.3 Multi-NIC Scheduling

Current smartphones are equipped with multiple radio interfaces, each enabling communication with a different wireless standard: Bluetooth for very short range communication, WiFi for local-area communication, and GPRS/EDGE or EVDO, and recently 3G technologies such as HSDPA, for wide-area communication. Several researchers have developed systems to take advantage of the complementary features of different networking technologies to save power on smartphones.

The Context-for-Wireless system [30] switches between the WLAN and the EDGE interfaces. 802.11 is preferred, but when it is not available the cellular network is used. Similarly CoolSpots [27] switches between WiFi and Bluetooth to save power with minimal changes in the infrastructure.

The Cell2Notify system [1] addresses VoIP applications on smartphones in enterprise environments. In this system a GSM call with a reserved caller ID signals the smartphone to turn on its WLAN interface. This approach works in enterprise environments that WiFi access is almost guaranteed, therefore the authors do not deal with the scanning problem.

Most of the research related to Multi-NIC devices deal with special applications and situations. In a recent work Zaharia and Keshav [37] studied the general case of multi-NIC scheduling on smartphones. They propose a hill-climbing algorithm that finds the optimal scheduling with better running performance compared to other optimization techniques. However, their work assumes future knowledge of the environment and connectivity opportunities, therefore their scheduling algorithm cannot be implemented in real systems.

[1]http://reality.media.mit.edu/

2.4 Access Point Selection

In the context of opportunistic communication using WiFi access points, a practical problem is selecting the best available access point. This is becoming more and more of a concern as more 802.11 access points are deployed. Virgil [25] tries to address this problem. It senses all the available access points, quickly associates to each of them and after running a number of performance benchmarks picks the best. The scanning strategy proposed in this thesis can be used to complement Virgil.

2.5 Measurement Studies

Several measurement studies on opportunistic communications, including ad-hoc opportunistic contacts [20, 19] and opportunistic communication with the infrastructure [5, 22] exist in the literature.

Researchers who study the characteristics of ad-hoc opportunistic contacts are mostly interested in the inter-meeting time between mobile users. Although many theoretical studies and simulations suggest that the inter-meeting time should follow an exponential distribution, experiments [20, 19] show a power-law distribution. Recently Cai *et al.* [6] claimed that the discrepancy on the tail behavior of the empirical data and the theoretical results lies the *finite boundary* assumption. In an unbounded domain, simulations and theoretical analyses result in power-law distributions. This is a disappointing observation for the future of the systems that rely only on ad-hoc contacts between users for content dissemination and networking, because the long tail of the inter-meeting times implies unbounded expected delay for such communication systems.

The first study on infrastructure opportunistic communication that looks into the length of WiFi availability times is by McNett and Voelker [22]. They distribute 275 Windows Mobile PDAs among collage freshmen to study their mobility patterns. Among other statistical characteristics, they report a long tail distribution for the length of sessions in their dataset. However, they express uncertainty about this observation. They attribute it to the fact that many of their users stopped carrying the PDAs, and left them on their cradles at home. In my measurement study I avoid this problem by providing strong incentive for the users to carry the smartphones with them.

Bychkovsky *et al.* [5] study vehicular Internet access using in situ WiFi access pints. Despite a 5 second latency in their connectivity measurement tool, due to under-powered hardware, they observe a long tail distribution in the length of connectivity times. They observe a similar distribution in the wireless opportunity times, and also the inter-connectivity times. This study is not concerned with energy constrained mobile devices, therefore they do not study the impact of the observed long tailed distribution on power management and WLAN management strategies.

2.6 Mobility Prediction

Identifying significant locations and routes of mobile users is an effective way of providing context information to mobile applications and systems. Learning regular mobility patterns can help predicting future location/activity and thus make mobile applications "smarter."

Several papers in pervasive computing [16, 12, 35] highlight the value that location-aware services can offer mobile users. However, the first papers that focus on extracting and learning *significant locations* of a mobile user are by Marmasse et al. [21] and Ashbrook et al. [3]. Ashbrook and Starner analyze GPS traces off-line and evaluate the results of their algorithm with geographical maps and the human subjects' help. This work is followed by another paper that considers the problem for multiple users [4].

This initial work has been extended in two directions: first, using technologies other than GPS to provide location hints to the system. Kang et al. [13] use PlaceLab [18] output, and Laasonen et al. [17] use the currently associated GSM cell tower ID as a hint to the user's location. Second, several papers have proposed alternative algorithms for clustering location traces into significant *places* [26, 38]. Especially, Hariharan et al. [11] present a generalized approach.

2.6.1 Positioning Technologies

The three major positioning technologies that have been used by researchers are the Global Positioning System, PlaceLab, and wireless cell IDs. GPS is globally available with relatively high precision (a few meters), but it is not available indoors. Several techniques have been used to overcome this limitation. The ComMotion project [21] uses loss of signal to infer an indoor location. Ashbrook et al. use a similar technique in combination with a variant of the k-means clustering algorithm to find the places where the user spends considerable time.

Kang et al. [13] use PlaceLab to address the indoor and urban limitations of GPS. However, PlaceLab can only be used indoors and in specific locations such as university campuses. Cell IDs are available virtually everywhere in urban areas, and current triangulation techniques allow positioning with 50 to 100 meters accuracy. But these techniques require knowledge of the wireless infrastructure, that is not available to the researchers. Also most cell phones do not reveal all the visible tower IDs to applications. Considering these limitations, the currently associated cell ID can serve as a "landmark" hint [17, 32].

2.6.2 Location Extraction Algorithms

Finding significant locations of a mobile user is a clustering problem. If the current location of the user is sampled with fixed frequency, the resulting samples tend to cluster around the significant places and along the important routes that a user takes between these places. However, inaccurate location samples, missing samples, and other sources of error make the clustering task non-trivial.

To be more effective, algorithms for extracting significant locations use both time and location. Ashbrook et al. [3] define a place "as any logged GPS coordinate with an interval of time t between it and the previous point." They use a fixed value for t (10 minutes) and extract all the places in their dataset. Then they use a customized centroid-based clustering algorithm, with a parameter for the maximum cluster radius, to cluster places according to their locations into what they call "significant locations." They find an optimal value for the cluster radius through finding a "knee" in the graph of the number of locations versus cluster radius. This fixed radius is used throughout their analysis.

Kang et al. [13] use a more on-line approach to clustering. Similar to Ashbrook et al. they use both a temporal threshold and threshold on cluster radius. When a new location is reported, it is compared to the previous location. If they are farther than the spatial threshold, the new sample is considered to belong to a new cluster. If the time duration of a cluster is less than the time threshold it is discarded, otherwise it is reported as a significant location.

Hariharan et al. [11] propose a more sophisticated approach, that works for a general class of location representations. They require the locations to exist in a metric space. They also require a raw location data to include the time-stamp in addition to the location. They provide algorithms to extract the patterns from a dataset. They state that the extraction of "stays" from a location history depends on two scale parameters: time and spatial scales. They present an algorithm, that given a "roaming distance" and a "stay duration," finds all the stays in a dataset. It identifies contiguous sequences of raw points, that remain within the roaming distance for at least as long as the stay duration as a stay. They use an agglomerative (hierarchical) clustering algorithm to cluster stays to destinations. Having stays and destinations, they build a Hidden Markov Model [29] for the location history.

BreadCrumbs [24] is a system that uses the same techniques for location extraction and builds the Hidden Markov Model to predict future wireless opportunities. Unfortunately, the accuracy of the predictions drop below 20% beyond two minutes. This result motivated me to consider heuristic scanning strategies rather than running the optimal algorithm of Chapter 3 on the future prediction.

Chapter 3

System Modeling

In this chapter I present an mathematical modeling of the problem in Section 3.1 and its optimal solution in Section 3.2. In Section 3.3 I simplify the optimal algorithm and prove that it will yield the optimal solution for all practically useful cases.

3.1 System Definition

The formal model, defined in this section, is necessary for the algorithmic analysis of the problem. For the moment, I will make a strong assumption, that is of future knowledge. The following definitions help in formally modeling of the problem:

Definition 1. *A medium, $m(.)$, is a mapping (function) from time to the set $\{$available, unavailable$\}$, with a finite number of discontinuities in any time interval (t_1, t_2).*

Definition 2. *An interface is a variable with three possible values (states): $\{$connected, disconnected, off$\}$. It can be connected at time t iff $m(t) =$ available. All state transitions of the interface incur a non-zero fixed cost in either direction, as in Table 3.1.*

States	**Trans. Cost**
off-disconnected	C_1
disconnected-connected	C_2
off-connected	C_3

Table 3.1: Fixed costs of state transitions of an interface

Where $C_1 + C_2 = C_3$. The connected state has a running cost of r, and the running cost of the disconnected state is r', such that $r' < r$.

Definition 3. *A schedule is any possible assignment of interface states to time. A T-connected schedule is a schedule that keeps the interface connected for time duration T.*

Definition 4. *An optimal T-connected schedule is a T-connected schedule that has the lowest cost among all other T-connected schedules on the same medium.*

Definition 5. *A strategy is an algorithm that, given a time interval, (a, b), medium, m, and a value, T, produces a T-connected schedule, if such a schedule exists.*

3.2 The Optimal Strategy

I present a dynamic programming strategy and prove that it generates the optimal schedule. First, I translate the problem to a variant of the fractional knapsack problem.

I will demonstrate that finding an optimal T-connected schedule is equivalent to finding the optimal solution to the Alchemists' Knapsack problem:

Alchemists' Knapsack: The goal is to entirely fill a knapsack of fixed capacity, V, with minimum cost using a number of items b_1, b_2, b_3, ..., b_n, each with cost, c_i, and integer size, s_i. It is possible to use a fraction of an item but the total cost of the item should be paid. It is also possible for two adjacent items, such as b_i, and b_{i+1}, to combine and form a new compound item with size $s_i + s_{i+1}$ and cost $c' < c_i + c_{i+1}$. The new compound item may also combine with its left or right items and so on. However, not any two adjacent items necessarily merge.

Theorem 1. *Finding an optimal T-connected schedule is equivalent to filling an Alchemists' Knapsack.*

Proof. Knowing that $m(.)$ has a finite number of discontinuities, It can be assumed that $t_1, t_2, \ldots, t_k$ are all the points in time that the medium changes state. Without loss of generality, assume that $m(a) = \textit{unavailable}$ for all $a < t_1$, therefore (t_1, t_2), (t_3, t_4), ... are all the intervals that the medium is available. I name such intervals *availability blocks* or *blocks.* For a given instance of the interface scheduling problem, I can construct an instance of the Alchemists' Knapsack as follows:

- Knapsack capacity: T
- There is an item for each connectivity block, with the length of the block, l_i, as its size. The cost of the item is: $l_i r + 2C_3$
- The order of the items is the order of blocks in time
- Two items, b_i and b_{i+1}, merge if considering the time between the corresponding blocks, $l_{i,i+1}$, it holds that: $l_{i,i+1}r' + 2C_2 < 2C_3$ (In other words, the running cost of the short idle time between the two blocks is less than the cost of switching off and turning on again.)

For any solution to the interface scheduling problem there is an equivalent solution for the above Knapsack problem, and vice versa. Therefore, if solution S_{ak} is an optimal solution to the Knapsack problem (i.e., has the lowest cost among all other solutions), its equivalent is an optimal solution to the interface scheduling problem. $\square$

Let f_i^j be the cost of the optimal solution for a knapsack of size j, using the first i items. Thus f_n^T is the cost of the optimal solution to the problem. The cost of the optimal solution presents the following optimal substructure:

$$\begin{aligned} f_i^j \quad = \quad & min\{ \\ & f_{i-1}^j, \\ & f_{i-1}^{j-l_i} + c_i, \\ & min_{1 \le k \le i}\{f_k^{j-l_{k...i}} + c_{k...i}\} \\ & \} \end{aligned} \tag{3.1}$$

where $l_{k...i}$ is the sum of the size of items $b_k, b_{k+i}, \ldots, b_i$ and $c_{k...i}$ is the cost of the item resulting from merging $b_k, b_{k+i}, \ldots, b_i$. This optimal substructure suggest using a dynamic programming algorithm to find the optimal solution.

3.3 The Greedy Strategy

If the length of the time between consecutive availability blocks is never smaller than $(2C_3 - 2C_2)/r'$, then no two blocks can merge. In this case a simpler greedy algorithm produces the provable optimal strategy.

The *greedy* strategy sorts all the availability blocks according to their cost in increasing order, and generates a schedule based on the following simple algorithm. The running cost is the same per unit of time for any block, therefore with the fixed transition costs sorting blocks in increasing order of cost translates to sorting them in decreasing order of length. Thus, in practice, I can reduce the schedule computation time considerably from $O(n^3)$ for dynamic programming to $O(nlgn)$ for sorting/greedy.

Algorithm 1 Greedy algorithm

```
$sum = 0$
while $sum < T$ do
    pick the next unused interval $(t_i, t_{i+1})$, that has the lowest cost
    $sum = sum + t_{i+1} - t_i$
    add ($t_i$, connected) to schedule
    add ($t_{i+1}$, disconnected) to schedule
end while
return schedule
```

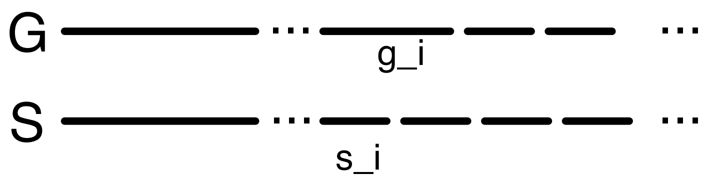

Figure 3.1: Availability blocks used by G and S sorted in decreasing order.

Theorem 2. *If no two blocks can merge, the greedy strategy always yields an optimal schedule.*

With the stated assumption, if I call the schedule returned by the greedy algorithm G, then for any other T-connected schedule such as S:

$$Cost(G) \leq Cost(S)$$

The following definition and lemma will help in proving the theorem:

Definition 6. *A complete schedule is one that fully utilizes all its availability blocks, except for at most one block. Otherwise the schedule is* incomplete.

Lemma 1. *For any T-connected incomplete schedule, S, there is a complete schedule, CS, that is T-connected and $Cost(SC) \leq Cost(S)$.*

To convert an incomplete T-connected schedule to its equivalent complete schedule, find all the blocks that are not fully utilized in the schedule and mark them as partial. Starting with any one, keep the interface connected throughout the entire availability block, remove its partial mark, and turn the interface off for the same amount of time on another partial block. Continue this until there is only one partial block left. These modifications do not increase the cost of the schedule.

Proof. Given a medium, assume there is a schedule, S such that $S \neq G$ and $Cost(S) < Cost(G)$. If S is not a complete schedule, then replace it with the equivalent complete schedule.

Lets sort the blocks in both G and S in decreasing length order as depicted in Figure 3.1, and label them. There exists a block such as s_i in S that $s_i \neq g_i$, otherwise $S = G$, therefore:

$$\begin{array}{lrcl} \forall j, 1 < j < i; & s_j & = & g_j \\ & s_i & < & g_i \\ \forall j, i < j \leq l = min\{|G|, |S|\}; & s_j & \leq & g_j \end{array}$$

Therefore

$$\sum_{j=1}^{i} s_j < \sum_{j=1}^{i} g_j$$

I assumed $\sum_{j=1}^{|S|} s_j = \sum_{j=1}^{|G|} g_j = T$, therefore $|S| > |G|$, which means $Cost(S) > Cost(G)$. $\square$

3.4 Summary

With a very strong assumption, that of knowing the future WiFi availability, the algorithm to find the optimal interface schedule is simple and has polynomial running time. In a special case, where the medium is never unavailable for short intervals, a simple greedy algorithm can find the optimal strategy.

There have been proposals and attempts to predict the future connectivity of a mobile user. In particular Nicholson and Noble [24] have proposed using a Hidden Markov Model to predict future connectivity of mobile users. Although they take advantage of locationing technologies, they can only reliably predict future connectivity for a few minutes (the accuracy of the prediction dramatically drops beyond two minutes). Therefore I will not follow this path of research. Instead I will investigate heuristic strategies for regularly scanning the environment. These strategies are introduced in the next chapter and briefly analyzed.

The optimal strategy is of little practical value. However, it provides a good reference for performance analysis of other WLAN management strategies as in done is Chapter 7

Chapter 4

Scanning Strategies

The only practical way for a smartphone to discover wireless connectivity opportunities in its environment is scanning. In this chapter, I assume there are background application processes that have pending data, and therefore need network connectivity, and to satisfy this requirement the smartphone operating system (or some other connectivity management software layer), should scan the environment regularly for possible connectivity opportunities.

In this chapter four heuristic algorithms for managing scans on the WLAN interface and simple analysis of each one are presented: Naïve Scanning, Static Scanning, Exponential Back-off, and Bounded Exponential Back-off Scanning. In addition, the possibility of using a cache of previous scan results is discussed.

4.1 Naïve Scanning

In the naïve approach, the smartphone continuously scans the environment until it finds a usable access point, and as soon as the connection is lost, aggressive scanning resumes. This strategy does not miss any connectivity opportunity; among all other scanning strategies Naïve Scanning has the lowest missed opportunity. However, this comes at a potentially high energy cost.

For a better analysis of the power consumption of the Naïve Scanning strategy, assume that each scan takes t seconds (for different smartphones this values varies between one and two seconds [2, 5]). The number of unfruitful scans (that do not find any opportunity) of this strategy is directly proportional to the amount of time that the user is not covered by any usable access point. Let λ be the ratio of the time that the user is not covered by WiFi to the total time of the day, then the number of scans during such a day is:

$$Scan_{Naive} \sim \frac{1}{t} \times \lambda \times 3600 \times 24 \qquad (4.1)$$

This value can potentially be rather large. For example if λ is 0.5, and each scan takes one second, then the smartphone will perform 43200 wasted scans each

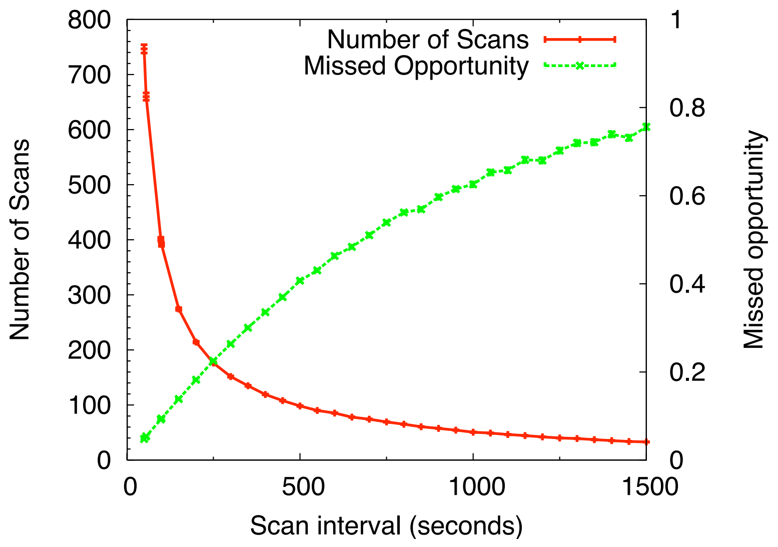

Figure 4.1: Number of scans and missed opportunity versus different values of scanning interval for one day derived by numerical simulation.

day. I present further evaluation of the performance of the Naive Scanning strategy using trace-driven simulation in Chapter 7.

4.2 Static Scanning

A simple extension to reduce the potentially high energy cost of the Naive Strategy, at the cost of some missed connectivity opportunity, is Static Scanning. If a scan does not find any usable access point, the next scan is delayed by a fixed amount of time, d, which I refer to as the scanning interval.

The parameter d can be used to adjust the performance of Static Scanning. Increasing d decreases the number of extra scans, but also increases the chance of missing opportunities. Similar to Naïve Scanning, the number of extra scans in one day is proportional to:

$$Scan_{Static} \sim \frac{1}{t+d} \times \lambda \times 3600 \times 24 \qquad (4.2)$$

The missed opportunity depends on the number (N), and the distribution of the length of the *availability* and *unavailability* blocks. Assuming $N = 100$ and exponential distributions for both availability and unavailability blocks with mean length $\gamma = \frac{0.6 \times 24 \times 3600}{N}$ and $\beta = \frac{0.4 \times 24 \times 3600}{N}$, I have used Monte Carlo simulation to find the number of scans and the missed opportunity. The result is plotted in Figure 4.1. This figure plots the number of scans and the missed opportunity time versus scan interval for $t = 2$, $\lambda = 0.4$, and $N = 100$. Similar results are obtained by varying t, λ, N.

$$\begin{tikzcd}
0 \arrow[r, bend left, "\lambda"] & 1 \arrow[r, bend left, "\lambda"] \arrow[l, bend left, "\mu"] & 2 \arrow[r, bend left, "\lambda"] \arrow[ll, bend left, "\mu"] & \cdots \arrow[r, bend left, "\lambda"] \arrow[lll, bend left, "\mu"] & k \arrow[r, bend left, "\lambda"] \arrow[llll, bend left, "\mu"] & \cdots \arrow[lllll, bend left, "\mu"]
\end{tikzcd}$$

Figure 4.2: Markov Chain model of the Exponential Back-off strategy. λ is the ratio of sum of unavailability time during some interval t to t, and $\mu = 1 - \lambda$.

With these assumptions, the Static Scanning strategy with relatively large scanning intervals can dramatically reduce the number of scans, but as seen in Figure 4.1, it comes at a large cost in terms of missed opportunity. Ideally, we would like to decrease the number of scans without increasing the number of missed opportunities. This motivates the next two heuristics

4.3 Exponential Back-off

I propose the Exponential Back-off strategy to improve the performance of Static Scanning in terms of missed opportunities. This strategy starts with a small value for the interval between consecutive scans, and increases this value exponentially. Therefore it is "adaptive", in the sense that several failures to find any opportunity is considered a sign of being out of range of wireless access, and as a result the scanning interval increases. Of course, if the interval increases to a large value, there is a risk that the device may completely skip over a short connection opportunity

Assuming an exponential distribution for the length of unavailability blocks (with rate λ), the Exponential Back-off strategy can be modeled with the Markov Chain in Figure 4.2.

The steady state equation is:

$$\forall k \geq 1 : \qquad \lambda P_{k-1} = \lambda P_k + \mu P_k = P_k \tag{4.3}$$

Knowing that:

$$\sum_{k=0}^{\infty} P_k = 1 \tag{4.4}$$

It can be derived that:

$$P_0 = \mu \tag{4.5}$$
$$P_k = \lambda^k \mu \tag{4.6}$$

If limited to one day, the number of back-offs cannot exceed $m = log(12 \times 3500)$. The expected number of scans in one day is:

$$\begin{aligned} E[Scan] &= \sum_{i=1}^{m} i \times P_i \\ &= \sum_{i=1}^{m} \mu i \lambda^i \end{aligned} \tag{4.7}$$

With the same assumptions underlying Figure 4.1, $\lambda = 0.4$, the total expected number of scans in a day is found to be 657. This corresponds to a scan interval of 56 seconds, when using the Static Scanning strategy.

The expected total missed opportunity can be evaluated as:

$$E[Missed_{EB}] = \sum_{i=1}^{m} P_i \times Missed_{static}(2^i) \tag{4.8}$$

Where $Missed_{static}(2^i)$ is the total missed opportunity of the Static Scanning strategy using intervals of length 2^i. Using Equation 4.6 and the numerical simulation results of Section 4.2, the expected missed opportunity of the example case is found to be 11 minutes. The Exponential Back-off strategy is achieving this by performing about 657. The Static Scanning strategy with a static interval of 56 seconds performs 659 scans and misses about 45 minutes. Therefore with these assumptions, the Exponential Back-off strategy does perform better than the Static strategy.

The expected length of scans performed by this strategy is:

$$\begin{aligned} E[d] &= \sum_{i=1}^{\infty} P_i \times d_0 2^i \\ &= d_0 \mu \sum_{i=1}^{\infty} (2\lambda)^i \\ &= \frac{d_0 \mu 2\lambda}{1 - 2\lambda} \end{aligned} \tag{4.9}$$

where d_0 is the length of the initial scanning interval. $E[d]$ is bounded only if $\lambda < \frac{1}{2}$. In other words the Exponential Back-off strategy enters a regime of very long scans if $\lambda \geq \frac{1}{2}$. In this case the maximum back-off time should be bounded by the scanning strategy.

4.3.1 Bounded Back-off

The scanning intervals, and consequently, the missed opportunity of the Exponential Back-off strategy can grow unbounded if $\lambda > \frac{1}{2}$. To solve this problem I

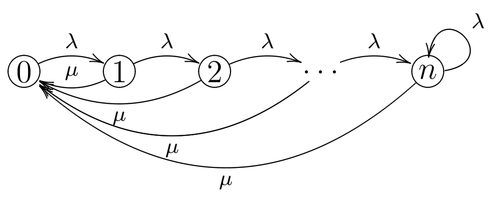

Figure 4.3: Markov Chain model of Exponential Back-off scanning with maximum back-off

propose bounding the back-off time to a maximum value corresponding to n successive back-offs. This modification also introduces a configurable parameter for the Exponential Back-off strategy. In the Markov Chain diagram of in Figure 4.3 the back-off time never increases beyond $d_0 2^n$.

Assuming a steady state system:

$$\forall k < n, k \geq 1 : \qquad P_k = \lambda^k P_0 \tag{4.10}$$

$$k = n : \qquad P_k = \frac{\lambda^n}{\mu} P_0 \tag{4.11}$$

Knowing that:

$$\sum_{k=0}^{n} P_i = 1 \tag{4.12}$$

It can be derived that:

$$P_0 = \frac{\mu}{2\lambda^n - 1} \tag{4.13}$$

Based on the discussions for the unbounded case, the expected value of the back-off time is:

$$\begin{aligned} E[d] &= \sum_{i=1}^{n} \mu P_i (d_0 2^i) \\ &= P_0 d_0 \mu [\sum_{i=1}^{n-1} (2\lambda)^i + \frac{(2\lambda)^n}{\mu}] \\ &= P_0 d_0 \mu [\frac{(2\lambda)^n - 2\lambda}{2\lambda - 1} + \frac{(2\lambda)^n}{\mu}] \end{aligned} \tag{4.14}$$

This value is bounded, regardless of the rate of the arrival of connectivity blocks.

4.4 Caching Scan Results

Another way of reducing the number of scans is using a cache of previous scan results. The cache stores a set of APs and their corresponding SSIDs and channels.

This technique can accompany any scanning algorithm. Whenever the scanning strategy requires a scan, instead of actually performing the scan, the interface can try to proactively associate with one or more of the APs stored in the cache. A similar technique has been suggested by Shin *et al.* [33] for reducing handoff latency. If none of the access points in the cache are usable, the interface can proceed with scanning. Like other caching schemes, a cache hit saves resources and a cache miss incurs extra cost. Therefore, the actual performance of a system that uses such a cache depends on the relative costs of hit and miss occasions and the cache hit ratio. The latter depends on the mobility of the user. I will study the performance of a novel caching technique in Chapter 7 using real user traces.

4.5 Summary

In this chapter I introduced four simple scanning strategies and roughly analyzed their performance with some assumptions about the environment of the smart-phone user. In all strategies there is a trade-off between the number of scans and the missed opportunity. The Naïve strategy is very aggressive in terms of scanning, therefore has the lowest missed opportunity at the cost of high energy consumption for scanning. The Static Scanning strategy can be adjusted, through the scanning interval parameter, to perform fewer scans, but at the cost of higher missed opportunity. The Exponential Back-off seems to have the advantages of both strategies. It starts aggressively but very quickly increases the interval between consecutive scans. If the rate of arrival of connectivity opportunities is low (less than $\frac{1}{2}$), the Exponential Back-off strategy enters a regime of very large scanning intervals, and therefore its missed opportunities grows unbounded. To solve this problem, I proposed setting a maximum on the back-off time. This chapter also raises the possibility of using a cache of previous scan results to reduce the number of actual scans performed by any scanning strategy. I used both analysis and numerical simulations to compare the behavior of these strategies assuming that unavailability block lengths were drawn from an exponential distribution.

Chapter 5

Measurements

The analysis of the strategies presented in Chapter 4 assume that unavailability blocks lengths are drawn from an exponential distribution. Although this eases analysis, it is not clear that this assumption holds true in practice. To validate that assumption and for a detailed and realistic evaluation of the performance of the proposed strategies, it seems appropriate to study real-world traces from mobile user.

I found a suitable dataset from a measurement carried out at Rice University [30]. To complement this dataset with traces collected with finer scanning granularity of scanning, and hopefully fewer missing samples, and also to cross-validate the results, I collected traces from six mobile user participants at the University of Waterloo.

In addition to the wireless experiment, I collected traces of the interaction of mobile users with their devices. To the best of my knowledge, this the first experiment that studies the statistical patterns of users' interaction with their smartphones.

In this chapter I present my measurement methodology in Section 5.1 and the dataset from Rice University in Section 5.2. I outline some statistical information about both datasets and compare them in Section 5.3 Finally Section 5.4 summarizes this chapter.

5.1 Measurement Methodology

I used iPhones [1] and BlackBerrys [2] with eight human subjects for the measurement study. This experiment was reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo under the ORE number 14849. This section presents details on the measurement methodology.

[1]http://www.apple.com/iphone/

[2]http://www.blackberry.com/

5.1.1 Wireless Scanning Experiment

Six iPhones with firmware version 1.1.4 were instrumented to scan on both the WLAN and GSM interfaces periodically. I used iPhones for this experiment for two reasons. First, it is one of the few smartphones that allows both WiFi and GSM scanning. Second, compared to the other smartphones that have this feature, such as those that use the Windows Mobile operating system, I found iPhone to be much more stable and reliable. I jailbroke [3] and unlocked the iPhones, and installed the GNU/BSD subsystem on them. Using the unofficial iPhone SDK, I developed two scanning tools: *TCLAPLogger* and *TCLGSMLogger*.

Each execution of TCLAPLogger commands the WLAN interface to perform an active scan on all the channels, and prints the following information about each discovered access point: ESSID [4], BSSID [5], a bit that is set to 1 if the access point is WEP enabled, and the signal strength.

Each execution of TCLGSMLogger commands the GSM interface to scan for the visible GSM cell towers. For each discovered tower it prints the following information: service Mobile Network Code ID, cell location ID, cell ID, and signal strength.

A daemon on the iPhones runs these two commands frequently and records their output along with a time stamp in a file on the local file system. The iPhone service start-up system was configured to launch this daemon on system start-up, and relaunch it if it dies for any reason.

In a pilot test I found that when the iPhone is not used for some time, it enters a deep sleep mode, in which all the system processes are stalled. To avoid this, I installed a third party application named *Insomnia*. It runs as a daemon and registers for all system power state change events. Whenever it receives a "kIOMessageCanSystemSleep" it calls "IOAllowPowerChange" with the proper parameters and therefore prevents the system from sleeping. The pilot test was also used to find a suitable value for the scanning interval. I found that any interval less than 60 seconds reduces the battery life to less than a day. Therefore one minute seemed the best scanning interval.

The six iPhones were given to six volunteers: two graduate students, two undergraduate students, a faculty member and a staff person. The volunteers were asked to use the iPhones as their cell phones (i.e., place their SIM cards in the iPhone). This provided enough incentives for the participants to keep the smartphones charged all the time and carry them with themselves. This experiment lasted for five weeks and at the end the logs were collected from the iPhone file system.

[3]Jailbreaking is the process that allows third party applications run on the iPhone

[4]Extended Service Set Identifier is a human readable string that identifies the wireless LAN.

[5]Broadcast Service Set Identifier is the MAC address of the station in the access point.

5.1.2 User Interaction Experiment

I used two BlackBerrys to log user interaction instead of iPhones for two reasons. First, they come with a data plan, therefore their users always enjoy Internet connectivity. Second, it turned out to be much easier to develop a logger application for user interactions on the BlackBerry compared to the iPhone.

This logging program is event-driven. Whenever the backlight turns on, as a result of user interaction with the keyboard, an event is sent to the logger by the BlackBerry OS. The logger also receives an event when the backlight turns off. This happens immediately after the smartphone is placed in the holster or a few seconds after it is not being used by the user. The daemon logs both of these events along with the current time in a file. The user logger was installed on the devices of two volunteer BlackBerry users.

5.2 Other Datasets

I found a measurement study similar to my wireless scanning measurement collected by Rahmati and Zhong [30] . They have made their traces available through the CRAWDAD Community Archive [15]. Although they have collected data using 14 Windows Mobile GSM smartphones, their dataset on CRAWDAD includes the data of only 10 subjects. Similarly to my study, they have scanned both WiFi access points and GSM cell towers every 60 seconds. I will refer to this dataset as the Rice dataset.

5.3 Characteristics of the Traces

In this section I present general statistics of the traces. The wireless traces have been filtered to include only open access points (i.e. not protected by wireless security such as WEP or WPA). I assume that all open access points are usable. This is not necessarily true, as many open access points redirect users to a "splash webpage" for authentication. However, I had no other way of finding usable access points without changing the nature of my wireless experiment. The Rice dataset does not include this information for the same reason.

5.3.1 Waterloo Wireless Dataset

The six subjects scanned 5709 unique WiFi access points over a period of five weeks. The number of scanned access points, and GSM cell towers for each user is summarized in Table 5.1 and plotted in bar charts in Figure 5.1. User 4 has visited considerably more WiFi and GSM IDs than all the other users, because she has

User	GSM Cell IDs	BSSIDs	APs per scan
1	721	1005	0.24
2	497	1363	1.78
3	233	1002	0.90
4	2287	2501	4.02
5	59	517	2.74
6	623	1158	1.32
Total	3070	5709	1.83

Table 5.1: Unique GSM IDs and BSSIDs visited by six subjects in the Waterloo dataset

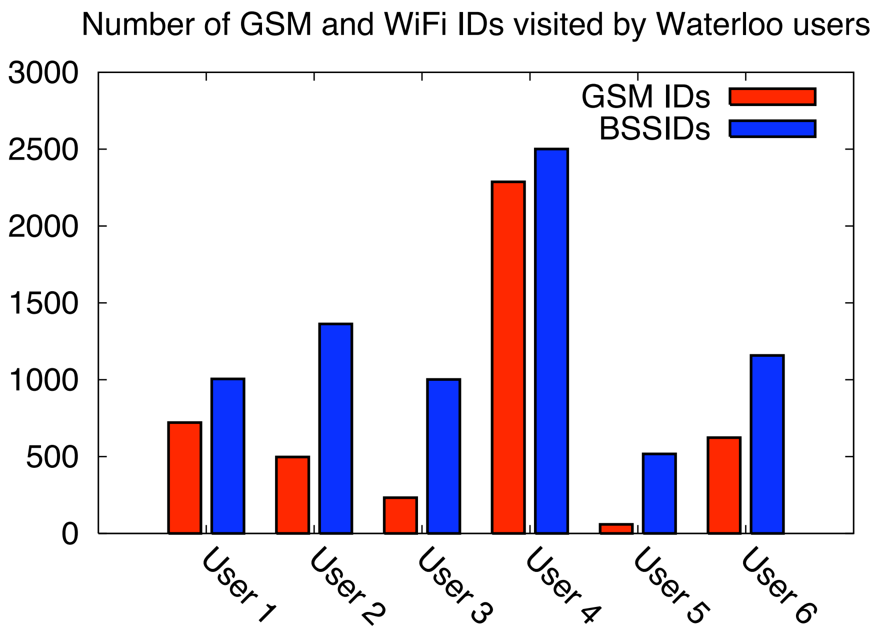

Figure 5.1: Comparison of the number of visited GSM and WiFi IDs by different Waterloo users. User 4 had two trips during the study and user 5 did not regularly carry the iPhone.

had two long trips (one out of the country). Also, unfortunately, user 5 did not carry the iPhone with him regularly.

Figure 5.2 visualizes a sample day of this dataset. Each GSM ID has been assigned a unique negative integer ID, and each BSSID has been assigned a positive integer greater than 10. For each sample point, each GSM ID is marked with a red plus and each BSSID is marked with a blue square. In this figure, the [1254, 1438] interval is a continuous WiFi availability block and [1204, 1253] is an unavailability block.

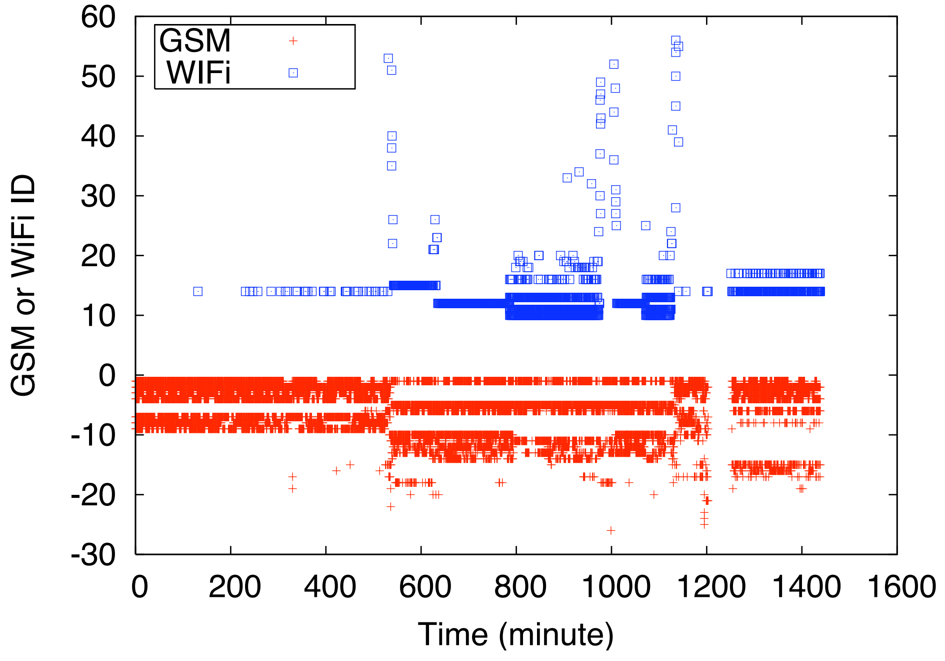

Figure 5.2: Visualization of a typical day from the Waterloo dataset

User	Availability$(1-\lambda)$	Blocks per day	Missing samples/day
1	0.14 ± 0.13	16 ± 18	41 ± 162
2	0.90 ± 0.20	43 ± 21	41 ± 163
3	0.61 ± 0.18	163 ± 76	87 ± 204
4	0.78 ± 0.24	143 ± 108	70 ± 196
5	0.90 ± 0.15	138 ± 93	115 ± 331
6	0.72 ± 0.21	135 ± 446	42 ± 164
Ave. ± Var.	0.62 ± 0.26	135 ± 55	66 ± 27

Table 5.2: Summary of availability information for the Waterloo dataset. Missing samples are treated as unavailable times

Table 5.2 summarizes the availability rate and the number of blocks for each user of the Waterloo dataset. User number 1 is a staff person who does not use wireless at home, and has the lowest availability rate among the users. Excluding this user from the dataset increases the average availability rate to 0.78. The mean length of an availability block in this dataset is 976 seconds (about 16 minutes).

Figure 5.3 plots the histogram of the length of the availability blocks with 60 second bins. The corresponding cumulative distribution function is plotted in Figure 5.4. In this Figure, inset is the CDF for blocks shorter than 1000 seconds.

The length of the availability blocks seem to follow a power-law distribution. This is further suggested by the log-log plots of the cumulative distribution function

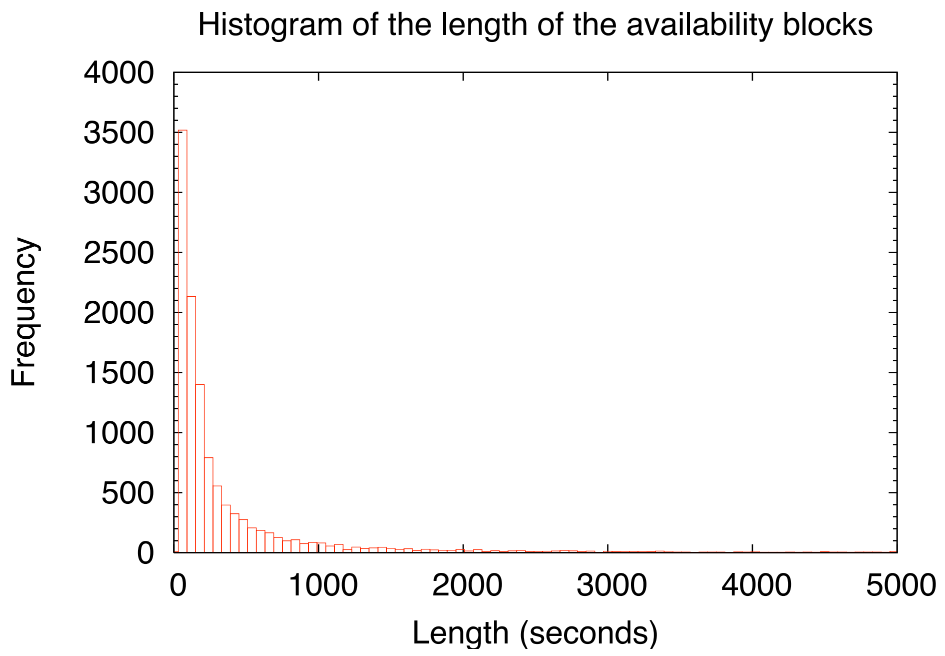

Figure 5.3: The histogram of the length of the availability blocks in the Waterloo dataset with bin size of 60 seconds.

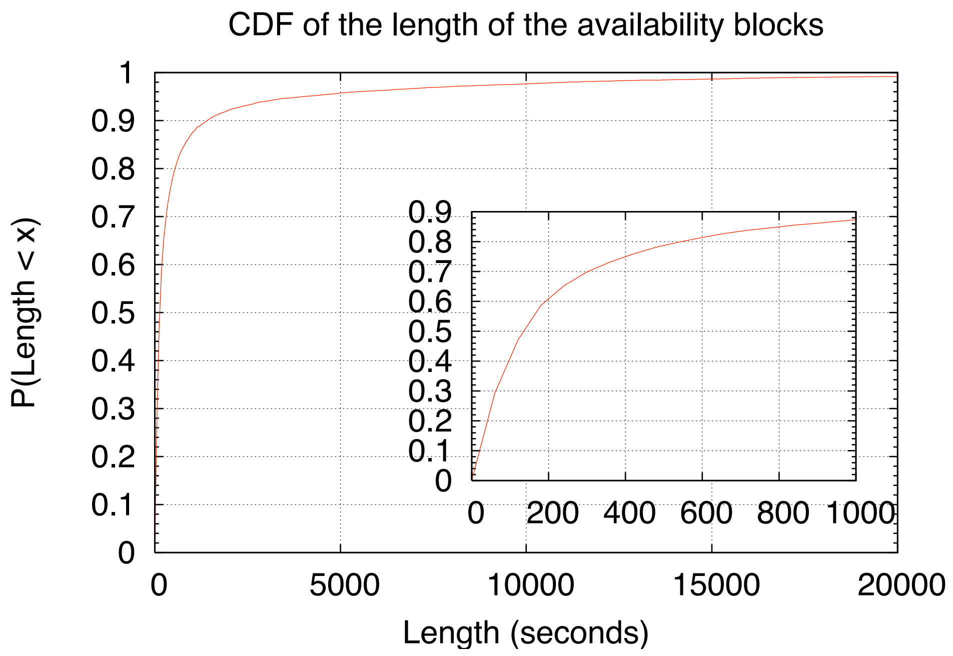

Figure 5.4: Cumulative distribution function of the length of the availability blocks of the Waterloo dataset. Inset is the CDF of block shorter than 1000 seconds.

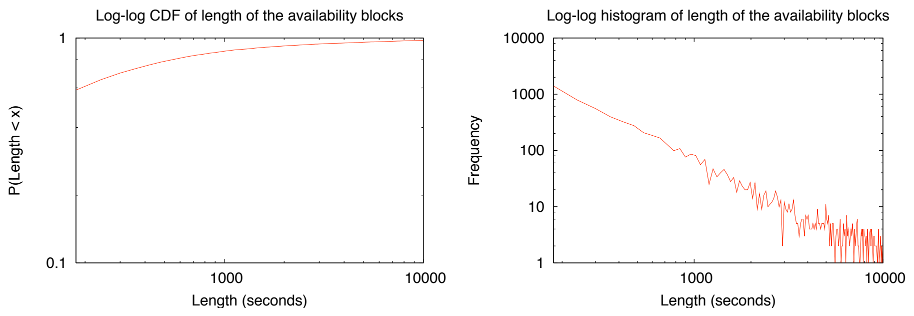

Figure 5.5: Log-log plot of the cumulative distribution function (left) and histogram (right) of the length of the availability blocks of the Waterloo dataset.

and histogram in Figure 5.5. Fitting a power-law distribution to the histogram of length of availability blocks using the goodness-of-fit based method described by Clauset, *et al.* [8] results in $\alpha = 1.59$.

Figure 5.6 plots the CDF and Figure 5.7 plots the histogram of WiFi unavailability time blocks in the Waterloo dataset. The average unavailability time is 223 seconds (about four minutes), and the values seem to follow a power-law distribution, as further suggested by the linear log-log plots of Figure 5.8. Fitting a power-law distribution to the histogram of the length of unavailability blocks using the goodness-of-fit based method [8] results in $\alpha = 1.52$. This is the first study, to the best of my knowledge, that has studied the distribution of the length of availability and unavailability blocks for WiFi interfaces on smartphones.

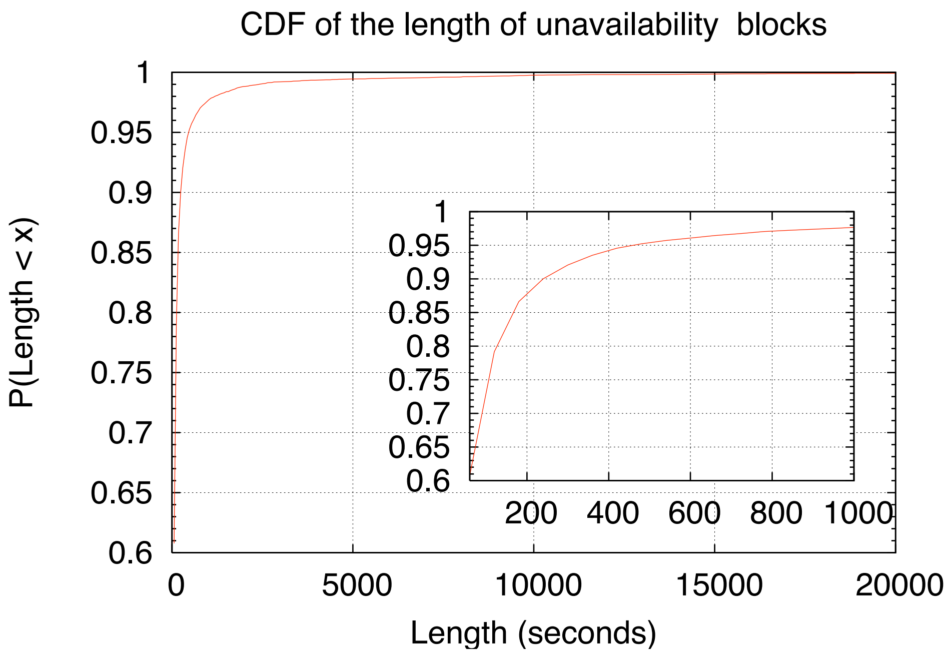

Figure 5.6: Cumulative distribution function of WiFi unavailability blocks of the Waterloo dataset

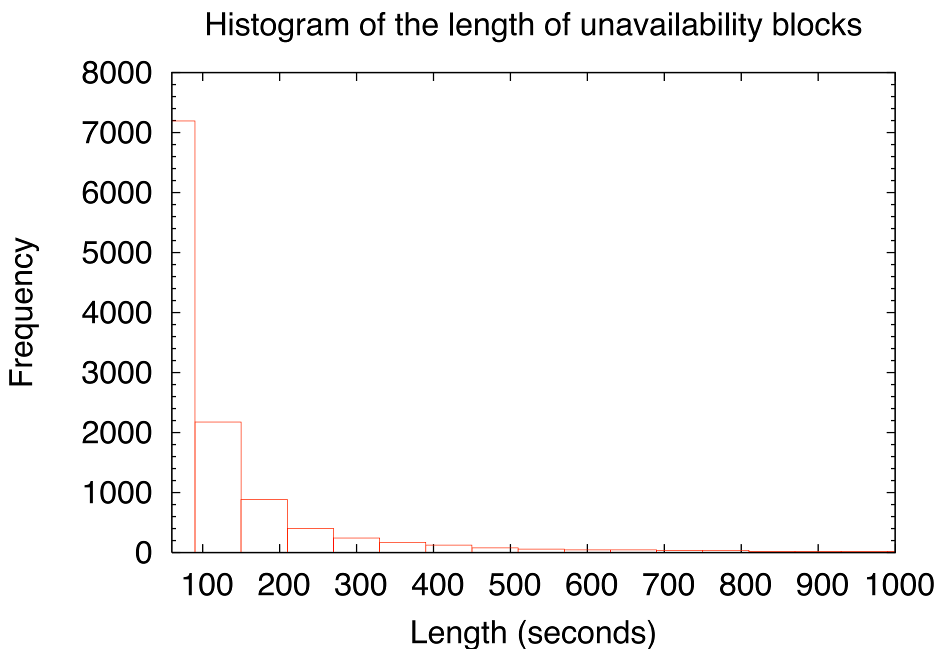

Figure 5.7: Histogram of WiFi unavailability blocks of the Waterloo dataset with 60 second bins

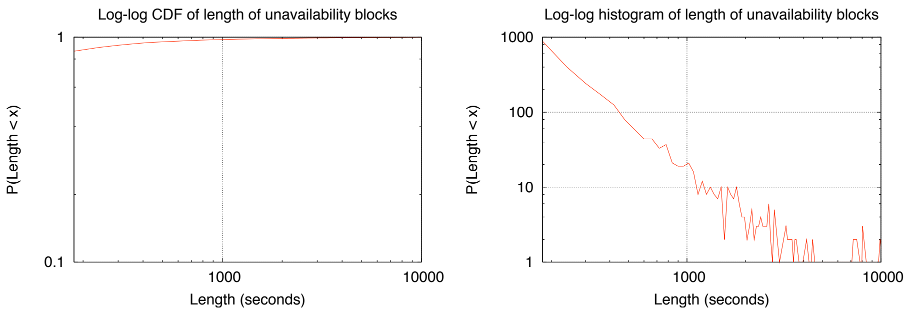

Figure 5.8: Log-log plot of the cumulative distribution function (left) and the histogram (right) of the unavailability time blocks of the Waterloo dataset.

User	GSM Cell IDs	BSSIDs	APs per scan
1	458	498	0.67
2	272	316	0.19
3	601	1088	1.23
4	310	774	1.71
5	284	258	1.24
6	747	567	2.38
7	954	876	1.43
8	771	580	2.54
9	292	457	0.55
10	636	858	1.59
Total	2806	3907	1.35

Table 5.3: Unique GSM IDs and ESSIDs visited in the Rice dataset

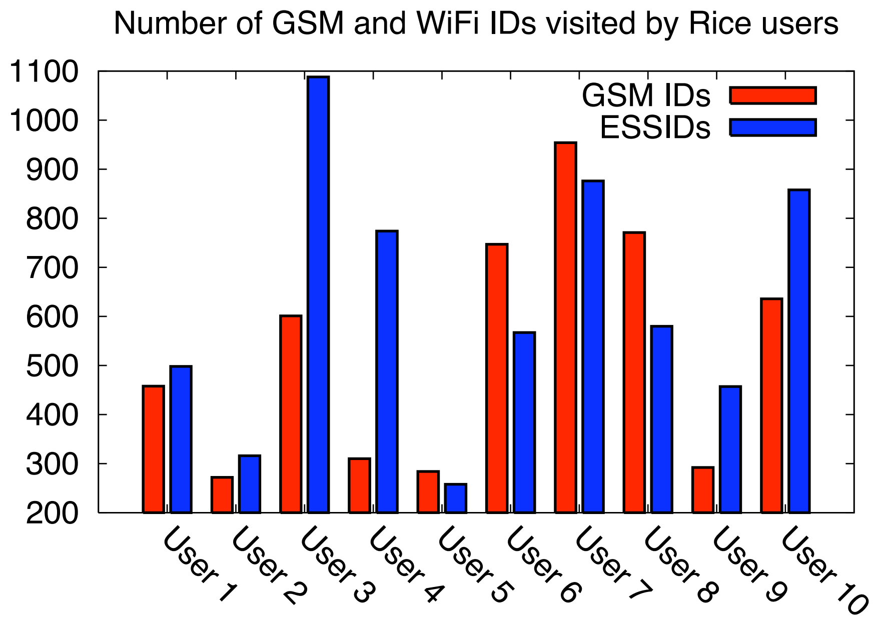

Figure 5.9: Comparison of the number of visited GSM and WiFi IDs by different Rice users.

5.3.2 Rice Wireless Dataset

In the Rice dataset 10 subjects scanned 2806 unique WiFi access points throughout four weeks between January and February of 2007. Table 5.3 summarizes the number of WiFi access points and GSM cell towers that each user in this dataset has scanned. Figure 5.9 presents the bar chart of those values for comparison.

The average number of missing samples for each day in the Rice dataset is more

User	**Availability**$(1-\lambda)$	**Blocks per day**	**Missing samples/day**
1	0.70 ± 0.14	10 ± 5	126 ± 369
2	0.81 ± 0.09	24 ± 25	181 ± 299
3	0.41 ± 0.15	6 ± 3	144 ± 397
4	0.40 ± 0.09	4 ± 2	74 ± 307
5	0.67 ± 0.15	180 ± 73	133 ± 377
6	0.55 ± 0.15	8 ± 3	202 ± 457
7	0.53 ± 0.18	181 ± 90	196 ± 408
8	0.61 ± 0.17	23 ± 36	88 ± 312
9	0.33 ± 0.25	292 ± 128	240 ± 401
10	0.80 ± 0.11	46 ± 50	87 ± 199
Ave. $\pm$ Var.	0.48 ± 0.17	77 ± 96	147 ± 53

Table 5.4: Summary of availability information for the Rice dataset. Missing samples are not taken into account for computing the availability rate.

than twice that of the Waterloo dataset. As summarized in Table 5.4, the average availability rate of WiFi in the Rice dataset and the average number of availability blocks per day are lower than the Waterloo.

Figure 5.10 visualizes a sample day of the Rice dataset. The effect of driving (i.e., traveling at high speeds) is evident in this figure at times 600, 800, 1000, and 1200 minutes into the day.

Despite these differences, the statistical properties of the Rice dataset are similar to the Waterloo dataset, and the length of availability blocks and unavailability blocks of both appear to follow power-law distributions.

Figures 5.11 and 5.12 plot the cumulative distribution function and the histogram of the length of availability blocks, respectively, and Figure 5.13 presents the corresponding log-log plots. Fitting a power-law distribution to the histogram of length of availability blocks using the goodness-of-fit based method [8] results in $\alpha = 1.66$, which is very close to the 1.59 value of α in the Waterloo dataset.

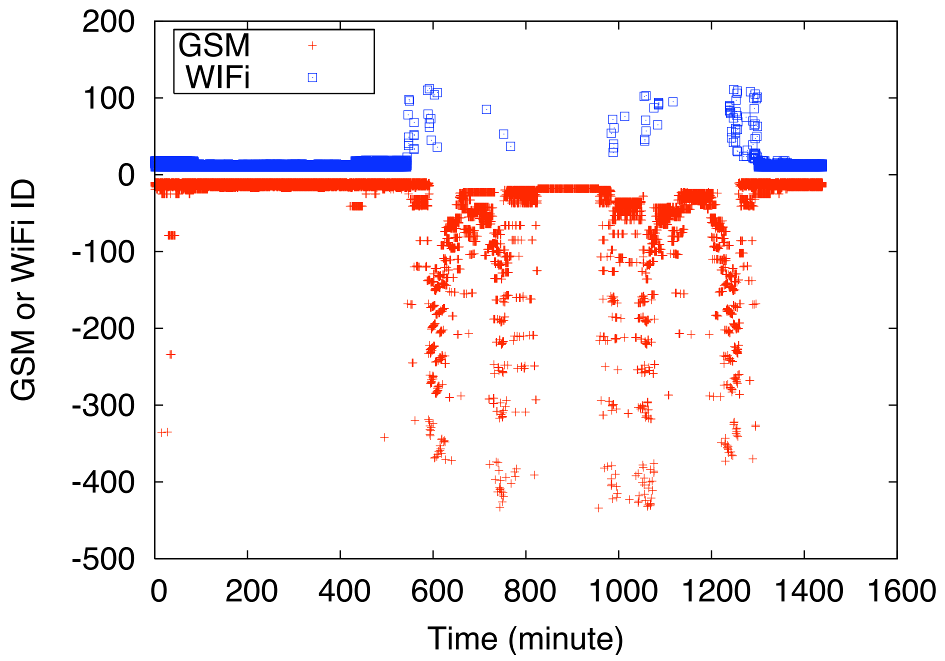

Figure 5.10: Visualization of a typical day from the Rice dataset. From 600 to 1300 minutes into the day five driving trips are visible.

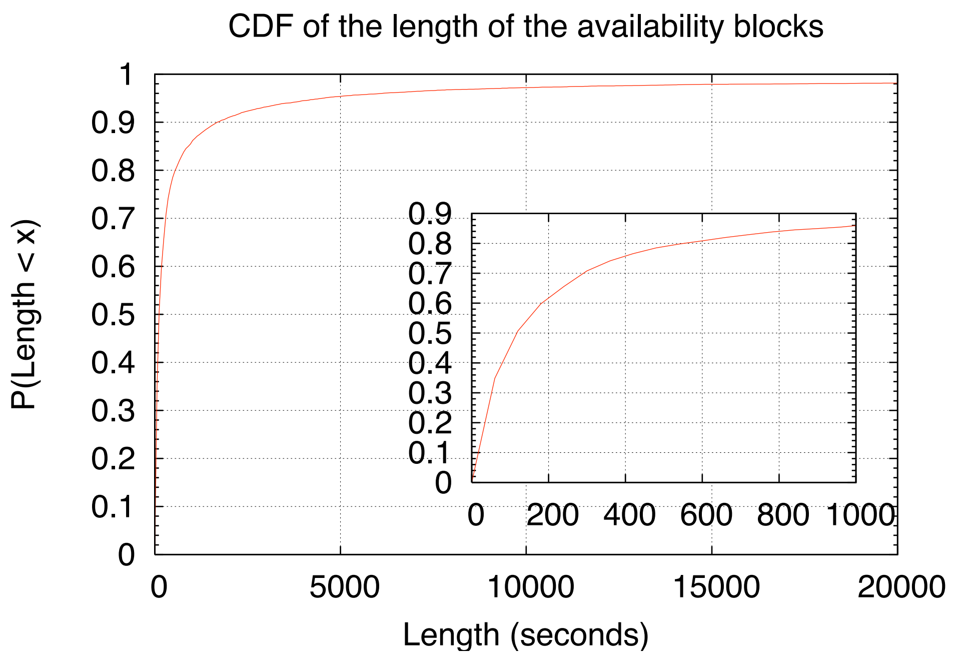

Figure 5.11: Cumulative distribution function of the length of the availability blocks of the Rice dataset. Inset is the CDF of block shorter than 1000 seconds.

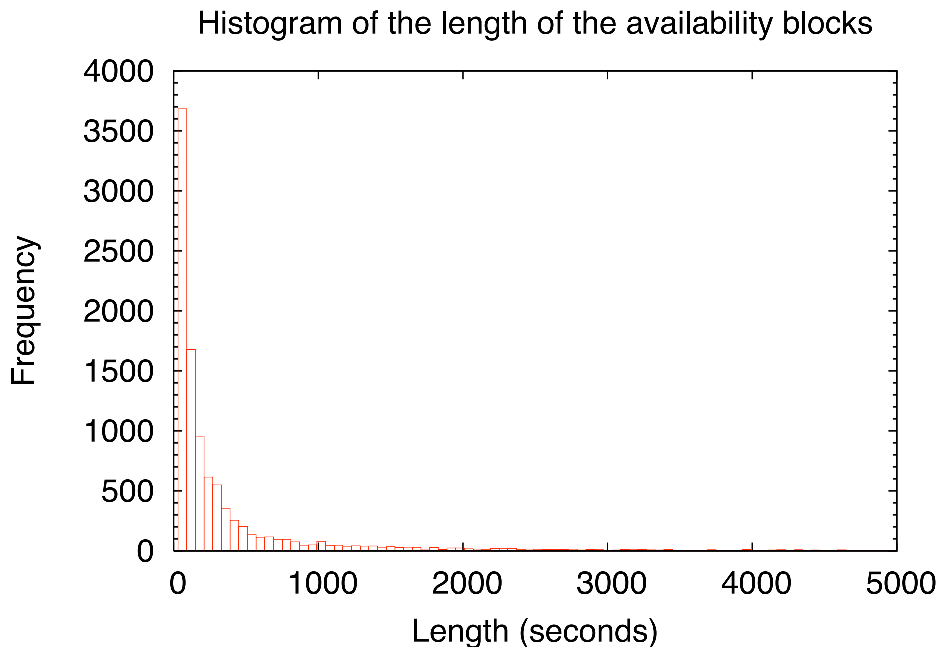

Figure 5.12: Histogram of the length of the availability blocks of the Rice dataset with 60 second bins.

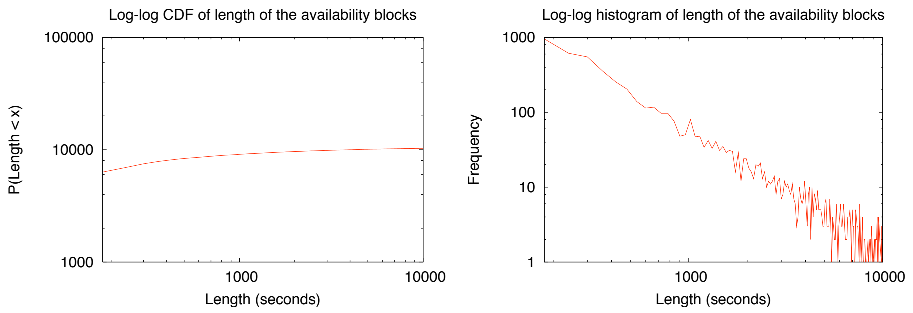

Figure 5.13: Log-log plot of the cumulative distribution function (left) and histogram (right) of the length of the availability blocks of the Rice dataset.

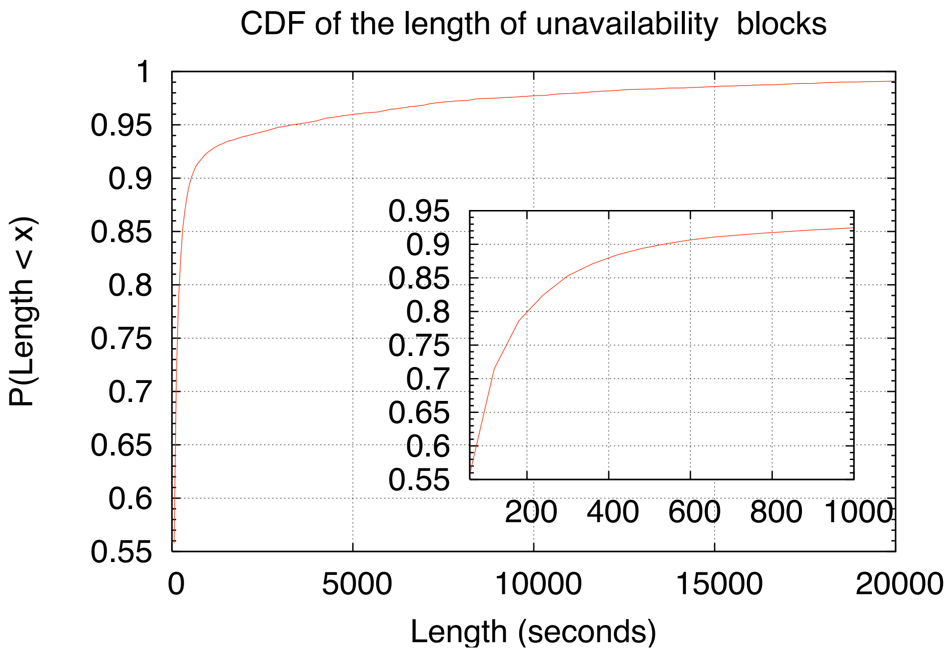

Figure 5.14: Cumulative distribution function of WiFi unavailability blocks of the Rice dataset. Inset is the CDF for unavailability blocks shorter than 1000 seconds.

Figures 5.14 and 5.15 plot the cumulative distribution function and histogram of WiFi unavailability blocks of the Rice dataset, respectively. Figure 5.16 presents the log-log plots. Fitting a power-law distribution to the histogram of length of unavailability blocks using the goodness-of-fit based method [8] results in $\alpha = 1.54$ that is close to the α of the unavailability blocks of Waterloo dataset, which is 1.52.

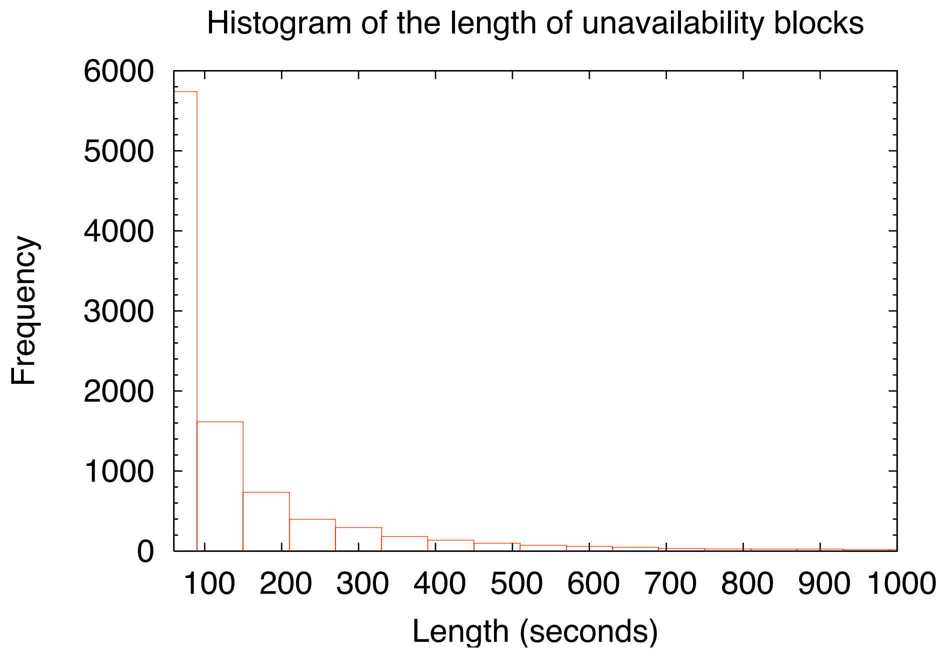

Figure 5.15: Histogram of WiFi unavailability blocks of the Rice dataset with 60 second bins.

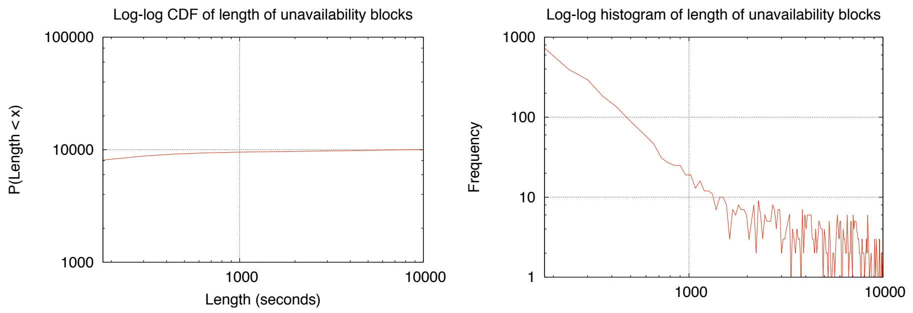

Figure 5.16: Log-log plot of the cumulative distribution function (left) and the histogram (right) of the unavailability time blocks of the Rice dataset.

User	Days	Interactions per Day	Mean Activity	Mean Inactivity
1	12	111 ± 64	40.68 ± 48.91	526.66 ± 2751.37
2	25	102 ± 49	52.68 ± 65.67	754.15 ± 2529.35
total	37	110 ± 55	48.19 ± 60.22	668.86 ± 2616.96

Table 5.5: General statistics about user interactions with their smartphones. The time durations are in seconds.

5.3.3 Waterloo User Interaction Dataset

I conducted this study with two other users, who carried Blackberry smartphones with them during the duration of the study. One of the participants of this experiment, referred to as user 1, quit the experiment after two weeks and the second participant continued for an additional week. Table 5.5 summarizes high-level statistics of the results.

Figure 5.17 plots the CDF and Figure 5.18 plots the histogram of the length of inactivity times for both users. The inactivity times seem to be from a power-law distribution. This is evident in the log-log plot of the CDF in Figure 5.19. The linear shape of the signature function strongly suggests that the length of inactivity intervals have a power-law distribution.

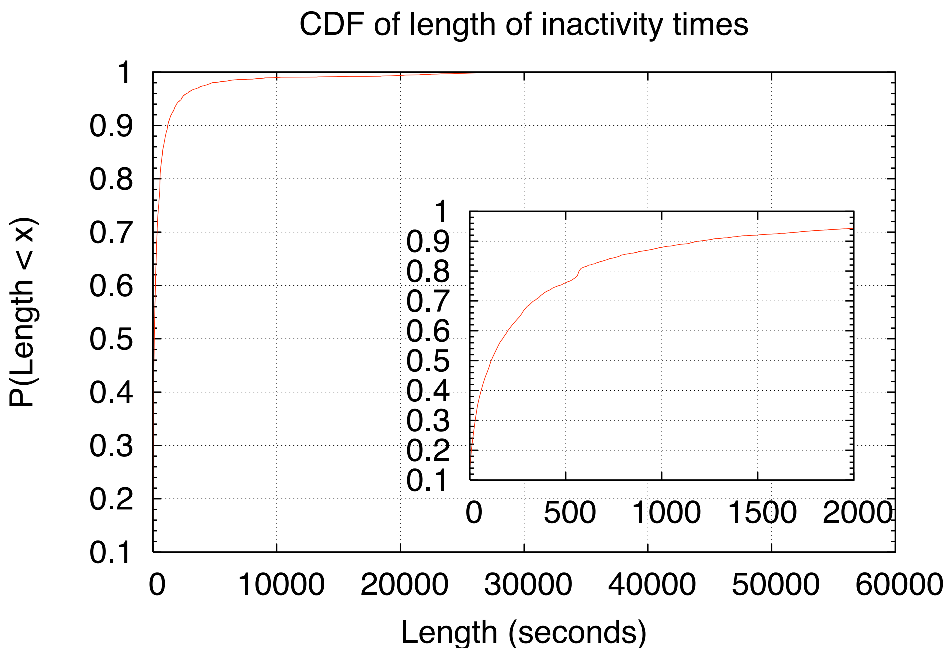

Figure 5.17: Cumulative distribution function of the length of user inactivity times in the Waterloo user dataset.

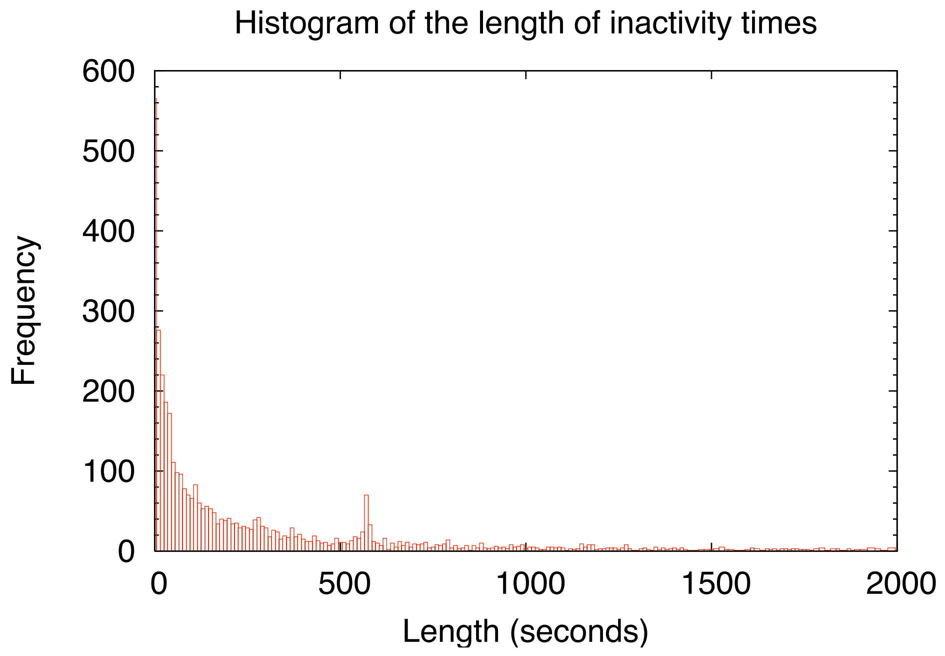

Figure 5.18: Histogram of the length of user inactivity times in the Waterloo user dataset with 10 second bins.

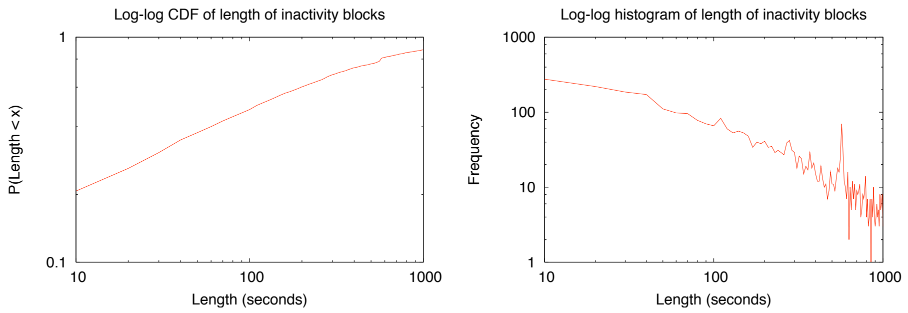

Figure 5.19: Log-log plot of the cumulative distribution function (left), and histogram (right) of the length of the inactivity times.

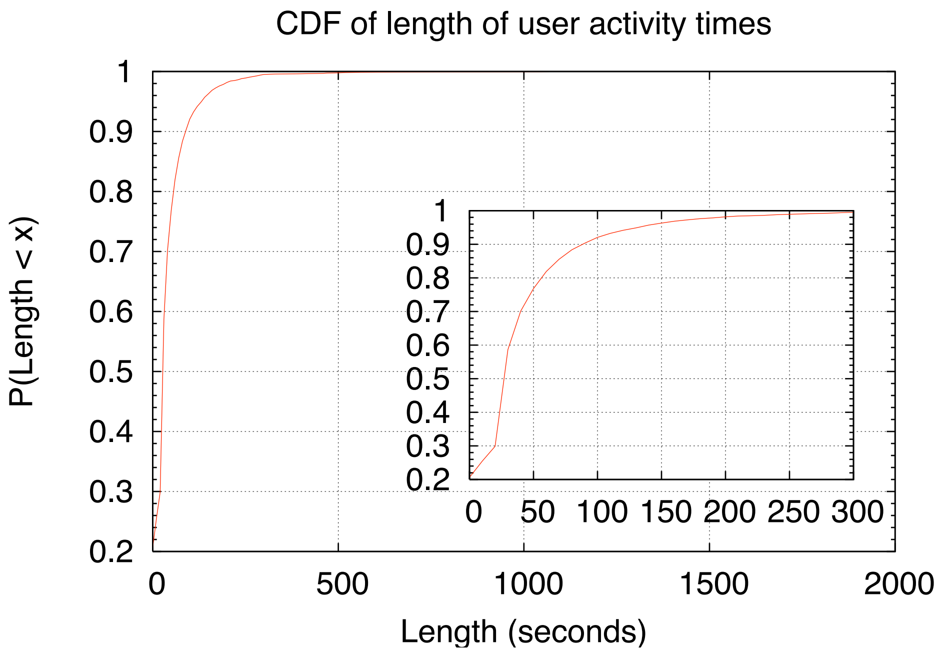

Figure 5.20: Cumulative distribution function of user activity times in the Waterloo user dataset.

Figures 5.20 and 5.21 plot the cumulative distribution function and the histogram of user activity times respectively.

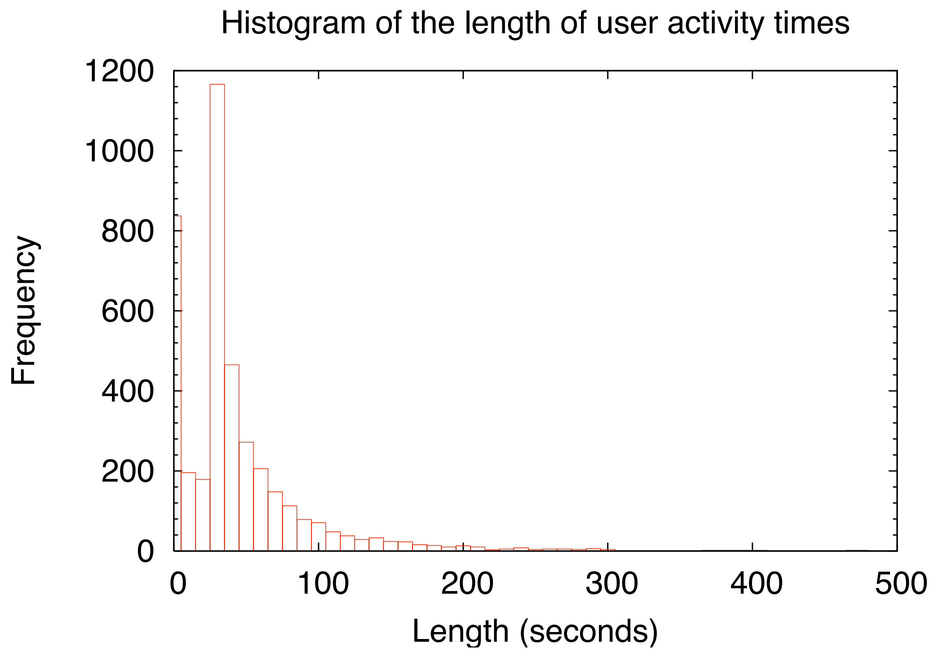

Figure 5.21: Histogram of of user activity times in the Waterloo user dataset.

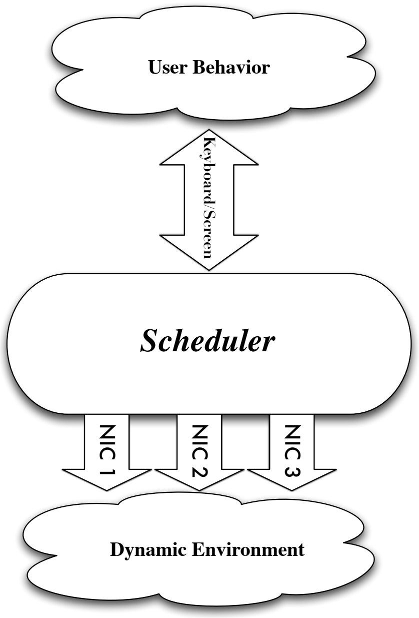

Figure 5.22: The scheduling algorithm for WLAN scanning has to deal with two sources of uncertainty: the user, and the environment

5.4 Summary

In this chapter I discussed the methodology of my experiments with the two dynamic environments of a mobile device. As depicted in Figure 5.22 the scheduling algorithm that decides when to scan for APs on the WLAN interface has to deal with two dynamic systems: the mobile users' behavior and the RF environment. Both of these have non-deterministic behavior. I used traces to study the statistical characteristics of both of these sources of uncertainty.

For the RF environment I used an independently collected dataset to validate my results, and found that the same statistical distributions govern the length of wireless opportunity times and unavailability times of both datasets.

Also this is the first time that the statistical behavior of mobile users in terms of interaction with smartphones has been studied. The inactivity times and the length of interactions with the smartphones seem to follow a power-law distribution.

Unfortunately, the results of this chapter invalidate the analyses of Chapter 4. Analyzing systems with power-law arrival time distribution times is mathematically intractable. Therefore for further evaluation of the proposed scanning strategies I resort to simulation. The next chapter details the simulation environment that I used for this purpose and Chapter 7 presents the simulation results.

Chapter 6

Trace-based Simulation

Many researchers are trying to build communication systems and applications to leverage the potentials of ubiquitous mobile computers and smartphones. The best approach to achieve this goal is implementing the intended system and evaluating it on real devices. However, it is always helpful to evaluate algorithms and systems with simulators before deployments, especially when implementation and deployment is as difficult as it is on cell phones [14].

The Opportunistic Connectivity Management Simulator (OCMS) is an extensible platform that I developed to simulate the behavior of a mobile device at a system level with respect to different opportunistic connectivity management algorithms.

The rest of this chapter is laid out as follows: OCMS design is presented in Section 6.1. Section 6.2 serves as a guide to using OCMS to evaluate and analyze connectivity management algorithms. Section 6.3 provides details about the implementation of OCMS and how to extend it.

6.1 Design Overview

OCMS is a single-thread discrete event simulator to evaluate the performance of different interface management strategies using traces from real-world user measurements. A high level overview of the architecture of OCMS is presented in Figure 6.1. This section provides details of the design of each of the components in this figure.

6.1.1 System Components

The Opportunistic Connectivity Management Simulator consists of the following components: the configuration subsystem, the event queue, network interfaces, schedulers, and the logging subsystem. Further details about each component follow:

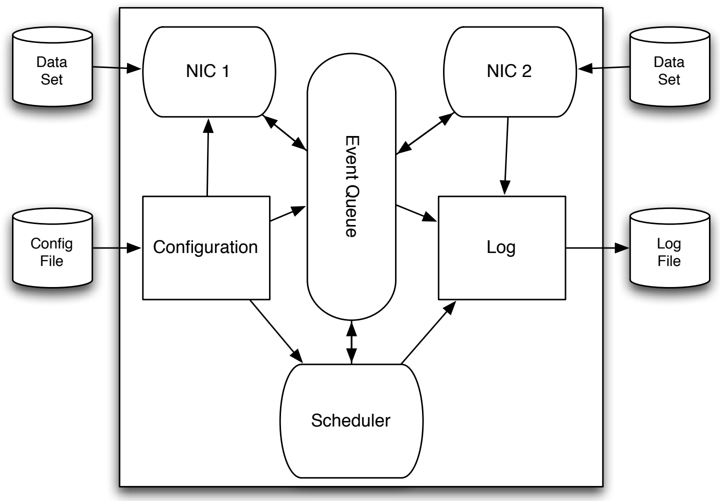

Figure 6.1: Design of the Opportunistic Connectivity Management Simulator

Configuration

An OCMS simulation starts by specifying the simulation configuration in a text file. This file is read by a configuration object, and its information is passed to the main simulator object and other components of the simulator. Further details of the configuration file are discussed in Section 6.2.

Event Queue

The heart of OCMS is an event queue. The components that are involved in a simulation communicate through this queue. The event queue finds the event with the lowest time and dispatches it to the destination. The semantics of an event is determined by the communication protocol between its sender and its receiver.

Network Interfaces

In OCMS, a mobile device can have an arbitrary number of network interfaces (NICs). As depicted in Figure 6.2 Each NIC has a *Medium*. A medium is a mapping from time to connectivity opportunities that can be provided in a file of real-world measurement traces, or generated based on a distribution function. Each NIC also has a number of *profiles* that keep track of different aspects of the NIC behavior, such as energy consumption.

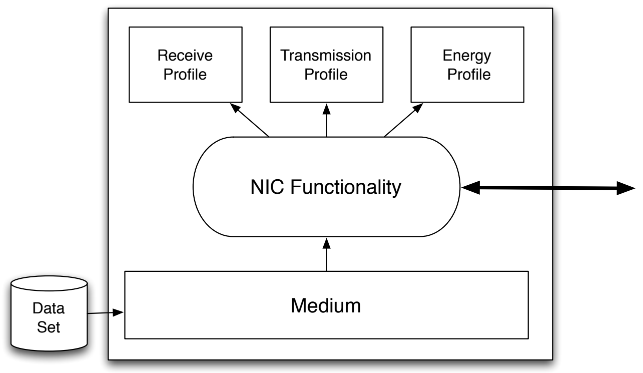

Figure 6.2: Internal structure of a network interface in OCMS

Schedulers

The intelligence of a mobile device is implemented in its scheduler (or schedulers). An OCMS simulation can have one or more scheduler objects. Each scheduler object interacts with one or more network interfaces through the event queue. Upon each query, the scheduler sends a command message to the sender of the query.

Logging

Different components of the simulation log their activities to a global *Logger*, which then filters them based on the desired log level and writes them in a log file. Each log entry has the following format:

```
log_level: sim_time  log_source log_message
```

6.2 Using OCMS

This section serves as a guide to using OCMS. If the desired network interface and scheduler are already implemented in OCMS, using them to run a simulation is as simple as writing a configuration file, otherwise the user can first implement the intended functionality and algorithm within the framework of OCMS.

6.2.1 Simulation Configuration File

The name of the simulation configuration file is passed as the only argument to the simulator executable. The configuration is a simple regular language. Each line of the configuration file consists of a *key* = *value* pair. The order of the pairs does not affect the simulation, but for better readability the file is divided into the following sections:

```
# Simulation start time (in seconds)
start_time = 300
# Simulation end time (in seconds). Zero means
# continue as long as the medium is available.
end_time = 0
# Log level. Possible values: OFF, INFO, DEBUG, PARANOID
log_level = INFO
# the log file name
logfile_name = sim.log
# disable standard output logs
STO = OFF
# dataset file
dataset = users/user1/3
# dataset time step in seconds
dataset_timestep = 60
# the special characters or ESSIDs to be filtered from the dataset
# you can specify multiple of them in different lines
dataset_filter = 0
# the frequency threshold of the high pass filter that is applied on
# data set (to filter out rare and random WiFi APs)
dataset_filter_highpass_freq = 0.1
```

Figure 6.3: Example of the simulation section of the OCMS configuration file

Simulation Section

This section of the configuration file contains all the general simulation parameters. An example is provided in Figure 6.3. This example specifies the start and end time of the simulation, the desired log level and the log file to be used. The file to be used as the source for traces and the time step of the samples. Simple filtering of specific symbols or based on frequency of the symbols can be specified in this section too; the simulator can filter symbols with very low frequency (i.e. random) from the dataset.

Interfaces Section

This section of the configuration file specifies the number of interfaces and different parameters of each interface. Each interface in OCMS has a name that will appear in the log messages generated by that interface, and is used to set its parameters in the configuration file. Each interface type has different configuration parameters, but all interface types require a medium that should get a dataset object.

```
# number of interfaces
interface_num = 2
# Name of the first interface.
interface_0 = wifi1
# The interface medium
wifi1_medium = dataset
# Type of the interface. Possible values: WIFI, GSM
wifi1_type = WIFI
# The dataset to be used as the medium of the interface
wifi1_medium = users/user1/0
# The following transition energy cost values are Integer.
# 'source-state'_to_'dest-state' = transition costs  (energy in millijoule)
wifi1_off_to_disc = 8000
wifi1_disc_to_off = 2000
wifi1_disc_to_con = 8000
wifi1_con_to_disc = 0
# Running costs are of the format:
# 'state-name'_run = running cost (power in milliwatt)
wifi1_disc_run = 100
wifi1_con_run = 150
wifi1_discscan_run = 675
wifi1_conscan_run = 675
wifi1_tx_run = 675
wifi1_rx_run = 500
# Running times for scanning states, the format is:
# ‘state_name’_time = running time (time in seconds)
wifi1_discscan_time = 2
wifi1_conscan_time = 2
wifi1_association_time = 1.5
# The second interface
interface_1 = gsm1
# type of the second interface is GSM
gsm1_type = GSM
gsm1_medium = dataset
```

Figure 6.4: Example of the interfaces section of the OCMS configuration file

Schedulers Section

The schedulers to be used in the simulation are specified in this section of the configuration file. Each scheduler may have specific parameters. Figure 6.5 is an example of this section of the configuration file.

```
# The type of the scheduler.
# Currently implemented schedulers are: OPTIMAL, STEPOPTIMAL, NAIVE, EB,
# LB, STATIC, USERSTATIC
scheduler_type = EB
EB_interface = wifi1
EB_maxbackoff = 600
# A zero value means transfer as much as possible
EB_goal = 0
```

Figure 6.5: Example of the scheduler section of the OCMS configuration file

6.3 OCMS Implementation

OCMS is implemented in Java, This section walks through the general structure of the implementation. However, interested readers are encouraged to find further implementation details in the OCMS Java API [1].

EventQueue and Event

An event is an object of class Event, that consists of a time-stamp, a source, a destination, type and a *details* array. The event queue, the single object of the *EventQueue* class, keeps all the events in a priority queue. In a tight loop it finds the next event with the lowest time and calls the *handleEvent* of its destination along with a reference to itself.

Any class that wishes to communicate through the event queue should implement the *EventConsumer* interface. In its handleEvent method, it can send replies to the sender of the event, through enqueueing a new event (or events) in the event queue.

To keep the implementation of other components of OCMS independent of the asynchronous communication nature of an event queue, I use wrapper classes to enable network interfaces and schedulers to talk to the event queue.

Network Interface

Each network interface class implements the *NIC* Java interface. A network interface, which is essentially a deterministic state machine, updates a number of *Profiles* every time it transitions to a new state. The current implementation of the WiFi network interface, for example, has three profiles: energy profile, transmission profile, and receive profile.

[1]http://blizzard.cs.uwaterloo.ca/mhfalaki/ocms/docs/

Scheduler

Scheduler classes implement the *Scheduler* interface. A Scheduler object has a 'query' method, that returns a NIC command whenever it is called. The command is one of the possible states of the network interface. The network interface tries to transition to that state and returns the result in the next query to the scheduler, until the end of simulation.

For example, if a network interface is in DISCONNECTED state, and the scheduler sends a NIC.CONNECTED command to the NIC object, the interface object tries to transition to the NIC.CONNECTED. If it succeeds, it will return NIC.CONNECTED, otherwise returns NIC.DISCONNECTED to the scheduler.

Medium, Experiment, and DataSet

Experiment is a general interface implemented by most components of OCMS that manage traces, most importantly the DataSet class. A DataSet object provides mechanisms for reading and manipulating time stamped samples from field experiments. A Medium object includes a DataSet object and provides an interface for network interfaces to access the dataset.

Log

OCMS log messages are generated by calling static methods of the *Log* class. Only classes, that implement the *Logger* interface, can use this logging facility. There is one static method for each log level and one other static method for standard output, therefore no other Java mechanism for output generation should be used.

6.4 Summary

This chapter introduced the Opportunistic Connectivity Management Simulator (OCMS), developed to study the performance of interface management strategies using real user traces. OCMS is implemented in Java and well documented [2]. It is designed to be extensible and usable by other researchers. The evaluations presented in next chapters of this thesis use OCMS and the traces introduced in Chapter 5.

[2]http://blizzard.cs.uwaterloo.ca/mhfalaki/ocms/docs

Chapter 7

Evaluation

In this chapter I present the results of the simulations performed with OCMS, introduced in Chapter 6, on the traces described in Chapter 5.

First I outline the simulation setting and configuration in Section 7.1. Using this OCMS configuration, I ran simulations to compare the energy consumption of the proposed scanning strategies, the results of which are presented in Section 7.2. Section 7.3 presents the effect of scanning interval in managing the trade-off between energy consumption and missed connectivity opportunity. Finally Section 7.4 concludes and summarizes this chapter.

7.1 Simulation Configuration

In the simulations that will be presented in this chapter, each day is simulated separately, therefore the simulation results refer either to a single day or to the average of all days for a user.

The OCMS configuration file specifies a data transmission goal for the scheduler. The scheduler uses the specified scheduling algorithm to keep the interface sufficiently connected to transfer the specified amount of data. After this point the scheduler turns the NIC off for the rest of the day to conserve energy.

After each simulation the log files are processed off-line to extract the desired performance metrics.

7.2 Comparing Strategies

In this section I compare the proposed scanning strategies in terms of the number of scans they perform to achieve a data transmission goal. Figure 7.1 compares the cumulative number of scans in a single day by different scanning strategies. The X axis is the total time that the interface is kept connected, and the Y axis is the

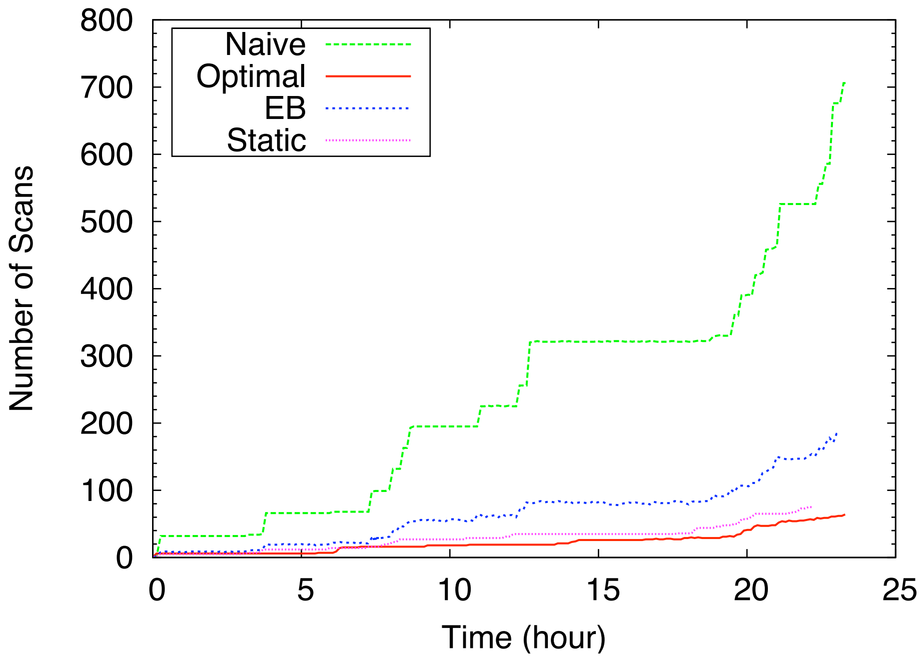

Figure 7.1: Comparing the cumulative number of scans performed by different scanning strategies for a typical day with 0.98 availability rate. The X axis is the total connected time during the day. The scanning interval of the Static strategy is five minutes and the Exponential Back-off (EB) is not bounded.

number of scans that the interface performs to achieve this goal. The scanning interval of the Static strategy is five minutes and the Exponential Back-off is not bounded.

As expected the Naïve strategy incurs a significant number of scans. However, the low number of scans performed by Static Scanning, which is almost as low as the optimal is surprising. Also contrary to my expectation, the Exponential Back-off strategy results in a larger number of scans compared to the Static strategy.

Figure 7.1 is for a day with a high connectivity rate (0.98). Figure 7.2 plots the number of scans of the same strategies on a day with much lower (0.47) WiFi availability rate. The Static Scanning strategy is again very close to the off-line optimal in terms of the number of scans. However, it is interesting to notice that among all the strategies Static Scanning has the lowest maximum connectivity time (about five hours).

To be able to draw general conclusions I ran the simulations on all the users' traces on both datasets and compared their average values.

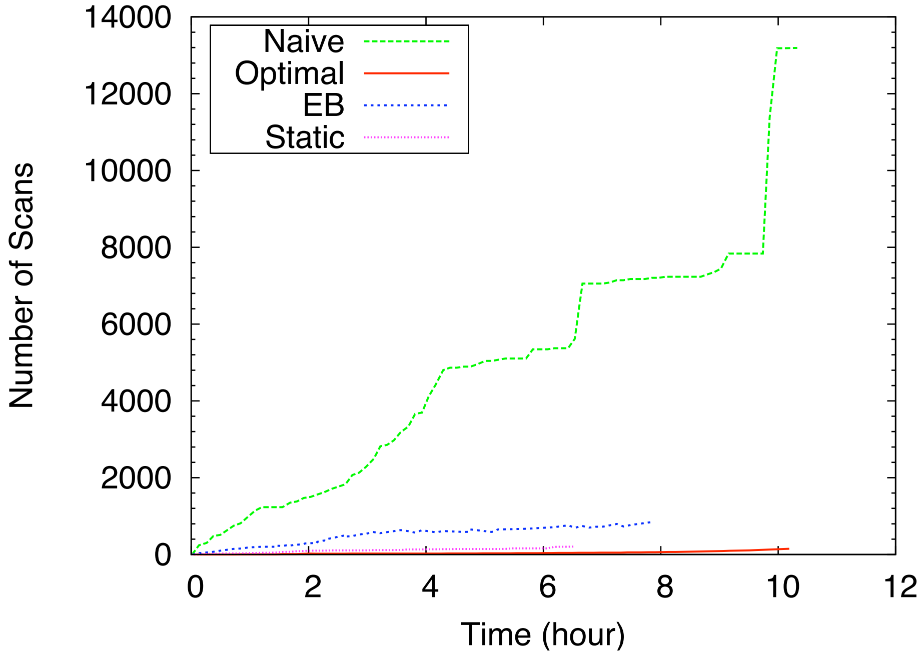

Figure 7.2: Comparing cumulative the number of scans resulted performed by different scanning strategies for a typical day with 0.47 availability rate. The X axis is the total connected time during the day. The scanning interval of the Static strategy is five minutes and the Exponential Back-off (EB) is not bounded.

7.2.1 Simulations on the Waterloo Dataset

Table 7.1 summarizes the number of scans performed by each strategy, trying to keep the interface connected as long as possible that is, assuming a very large data transmission goal. The Naïve strategy performs a considerably higher number of scans, and obviously should be avoided in any battery operated system. To compare the results of Static and Exponential Back-off strategies, the values in this table are plotted in the bar charts of Figure 7.3.

In the Waterloo dataset, except for user 1, the Static Scanning strategy performs considerably fewer scans compared to the Exponential Back-off strategy. In fact the number of scans of the Static strategy with an interval of five minutes is very close to the off-line optimal. However, reducing the number of scans comes at the cost of higher missed opportunity. Therefore, before drawing the conclusion that the Static strategy performs better than Exponential Back-off, the missed opportunity should be considered. Table 7.2 summarizes the ratio of missed wireless opportunity to the total available opportunity by each strategy for the Waterloo users. Availability is the ratio of the time use is covered by WiFi during each day to the length of the day. The blocks column presents the average number of availability blocks per day for each user.

Ignoring the variance caused by the length of the scan operation, the Naïve strat-

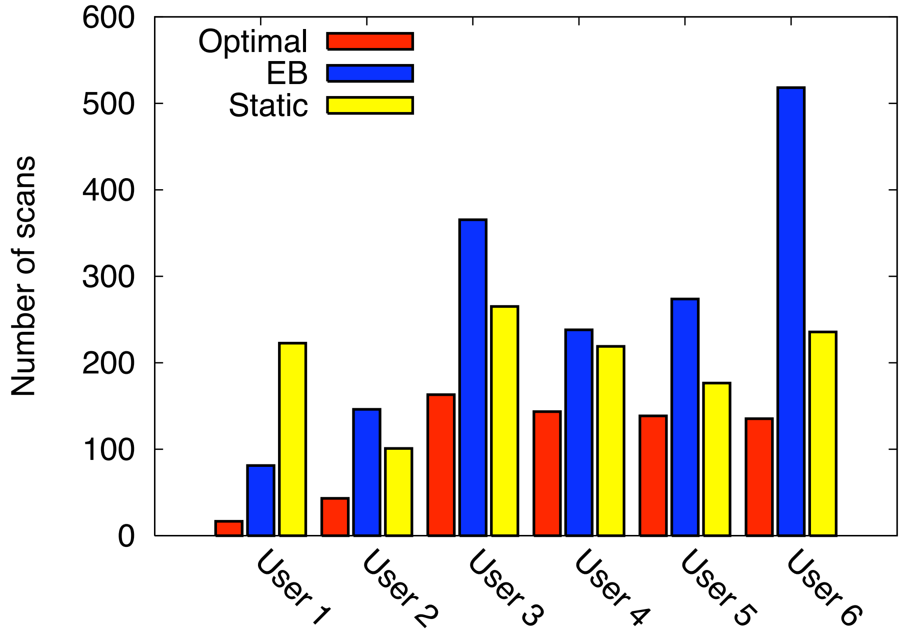

Figure 7.3: Comparing the number of scans performed by Optimal, Static, and Exponential Back-off strategies on the Waterloo dataset

egy does not miss any connectivity opportunity. The values of Table 7.2 for the Static and Exponential Back-off strategies are plotted in Figure 7.4. Based on Figures 7.3 and 7.4, I can conclude that Static scanning with a relatively large scanning interval (five minutes in this example) on average performs better than the Exponential Back-off strategy both in terms of the number of scans and missed opportunity. In Section 7.3 I will explore the effect of varying the length of the scanning interval.

user	**Availability**	**Optimal**	**Naïve**	**Static**	**EB**
1	0.14	16	31170	222	81
2	0.90	43	7459	100	146
3	0.61	163	20661	265	365
4	0.78	143	13423	218	238
5	0.90	138	6439	176	273
6	0.72	135	16712	235	518
Ave. $\pm$ Var.	0.62 ± 0.26	106 ± 55	15977 ± 8399.9	203 ± 52	270 ± 143

Table 7.1: Average number of scans performed by different scanning strategies for all the users of the Waterloo dataset.

user	Availability	Blocks	Static	EB
1	0.14	16	0.10	0.60
2	0.90	43	0.05	0.08
3	0.61	163	0.31	0.48
4	0.78	143	0.07	0.14
5	0.90	138	0.09	0.09
6	0.72	135	0.28	0.36
Ave. ± Var.	0.62 ± 0.26	106 ± 55	0.15 ± 0.10	0.29 ± 0.20

Table 7.2: Average ratio of missed opportunity to the total wireless opportunity for different scanning strategies for all the users of the Waterloo dataset.

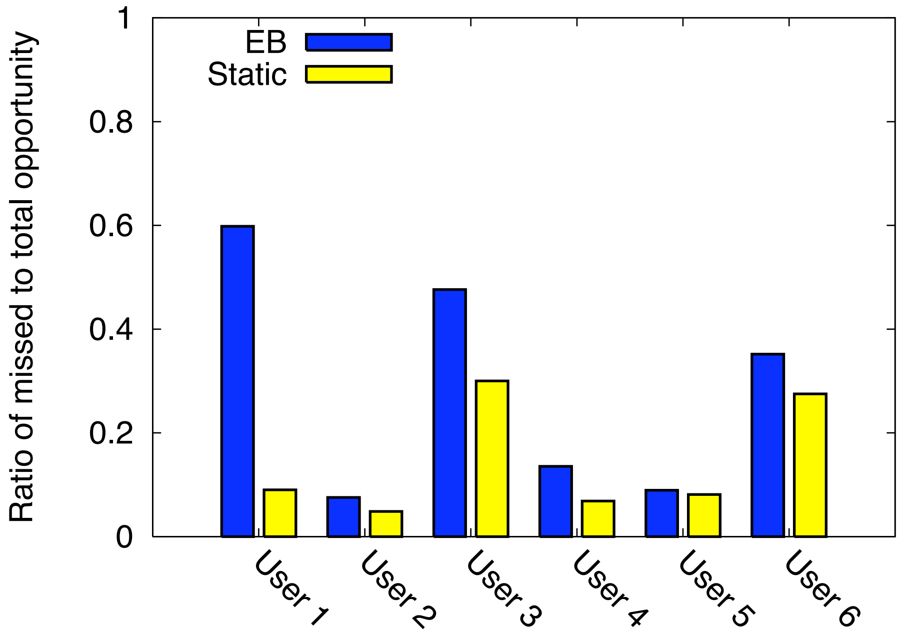

Figure 7.4: Comparing the missed opportunity by Static, and Exponential Back-off strategies on the Rice dataset.

7.2.2 Simulations on the Rice Dataset

Based on the results in Chapter 5, I expect to get similar results for the same simulations on the Rice dataset. Table 7.3 summarizes the average number of scans performed by the four strategies for each of the Rice dataset users, and Figure 7.5 plots the values for the Optimal, Static and Exponential Back-off strategies.

For the number of scans, four of the Rice users present similar results to the five Waterloo subjects. But for users 1, 2, 3, 4, 5, 6, and 8 of the Rice dataset, similar to user 1 of the Waterloo dataset, the Static strategy ends up performing considerably higher number of scans (about twice) compared to the Exponential

user	Availability	Optimal	Naïve	Static	EB
1	0.66	10	14392	106	47
2	0.14	24	26155	208	87
3	0.39	6	26126	179	33
4	0.37	4	27372	185	32
5	0.56	180	16470	277	268
6	0.55	8	20690	144	38
7	0.54	181	20042	299	317
8	0.60	23	19252	145	53
9	0.31	292	22342	356	406
10	0.76	46	11516	127	184
Ave. $\pm$ Var.	0.48 ± 0.17	77 ± 96	20436 ± 5002	167 ± 86	106 ± 137

Table 7.3: Average number of scans performed by different scanning strategies for all the users of the Rice dataset.

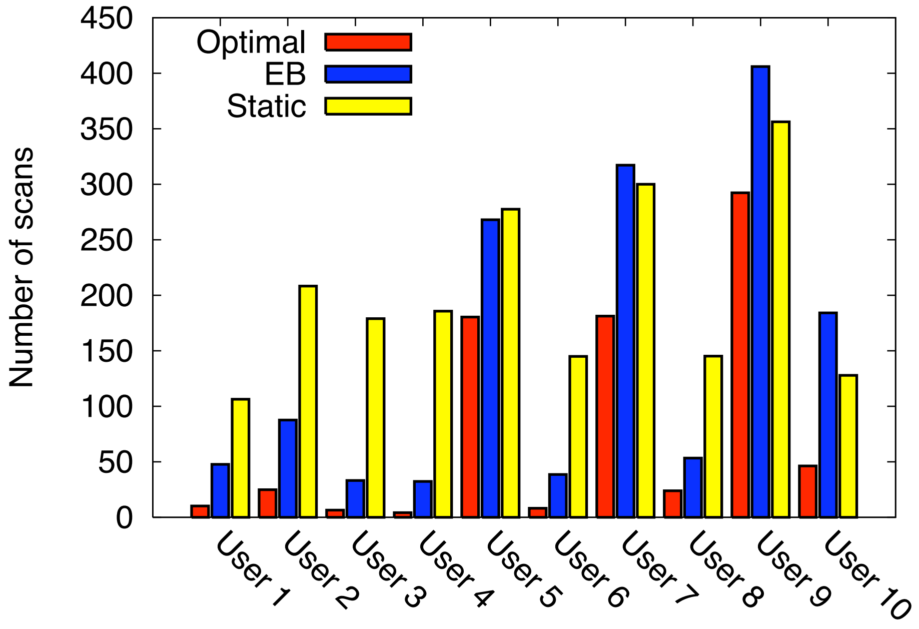

Figure 7.5: Comparing the number of scans performed by Optimal, Static, and Exponential Back-off strategies on the Rice dataset.

Back-off strategy.

But the missed opportunity of the Static strategy is consistently lower than the Exponential Back-off (about 30%) for all the users.

user	Availability	Blocks	Static	EB
1	0.66	10	0.03	0.22
2	0.14	24	0.21	0.74
3	0.39	6	0.02	0.18
4	0.37	4	0.01	0.23
5	0.56	180	0.14	0.27
6	0.55	8	0.02	0.12
7	0.54	181	0.14	0.19
8	0.60	23	0.04	0.19
9	0.31	292	0.45	0.75
10	0.76	46	0.09	0.14
Ave. $\pm$ Var.	0.48 ± 0.17	77 ± 96	0.11 ± 0.12	0.29 ± 0.20

Table 7.4: Average ratio of missed opportunity to the total opportunity for different scanning strategies on the Rice dataset.

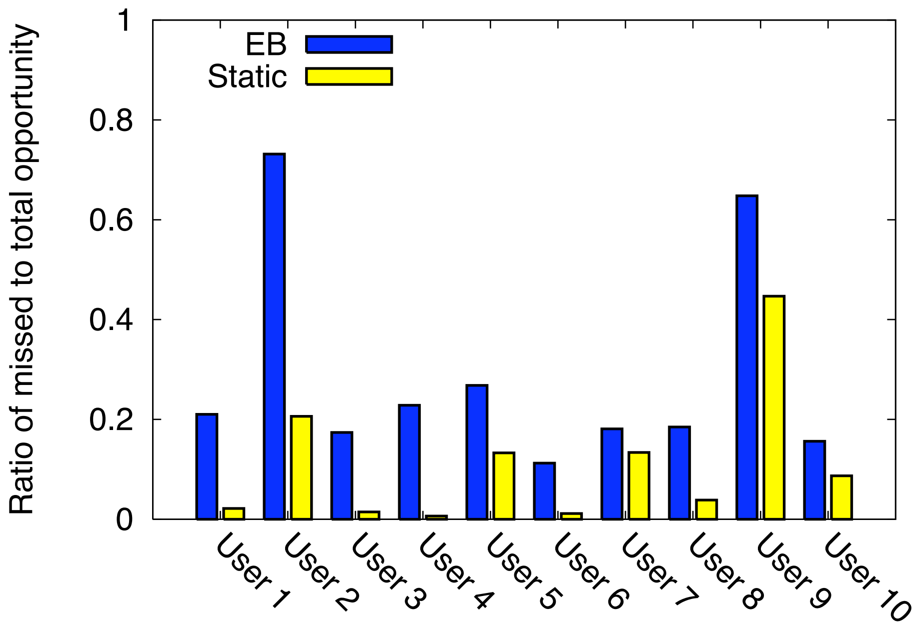

Figure 7.6: Comparing the captured opportunity by Static, and Exponential Back-off strategies on the Rice dataset.

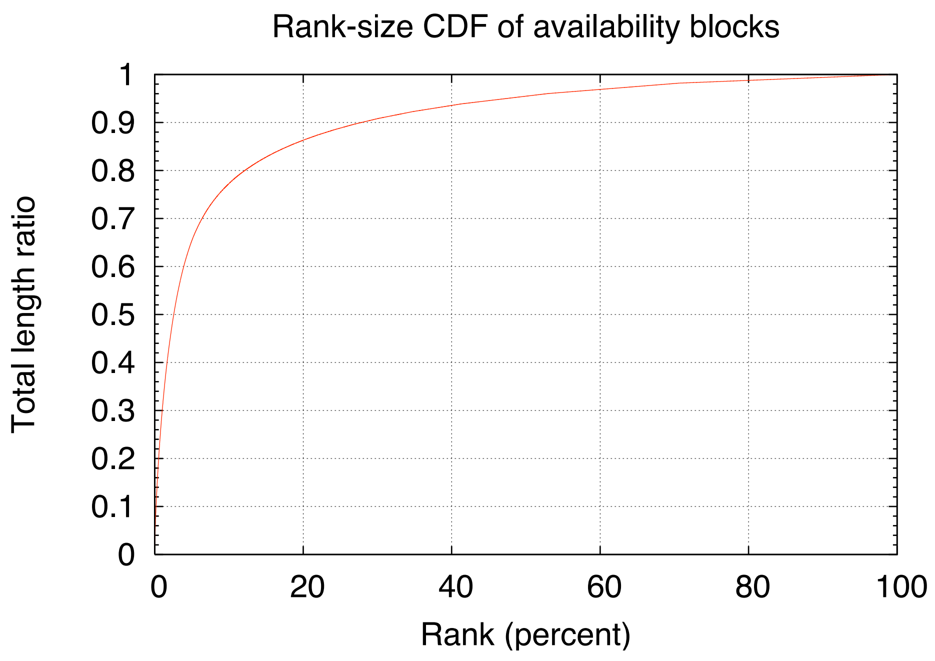

Figure 7.7: Rank-size cumulative distribution function of the length of availability blocks in the Waterloo dataset.

7.2.3 Discussion

In this section I present my justification for the following two observations:

- For most users, the Static strategy performs fewer scans and misses less opportunity compared to the Exponential Back-off strategy
- For some users Exponential Back-off performs fewer scans, but at a high missed opportunity cost

Static Scanning performs well

To explain this observation, I need to explain:

1. Why the Static strategy's missed opportunity is relatively low
2. Why the Static strategy performs few scans in most of the cases

I attribute the low missed opportunity of Static scanning (which is relatively true for Exponential Back-off, too) to the power-law distribution of the length of the WiFi availability blocks. The rank-size distribution of the WiFi availability blocks of the Waterloo dataset in Figure 7.7 illustrates the effect of the power-law distribution on the missed opportunity of a scanning strategy.

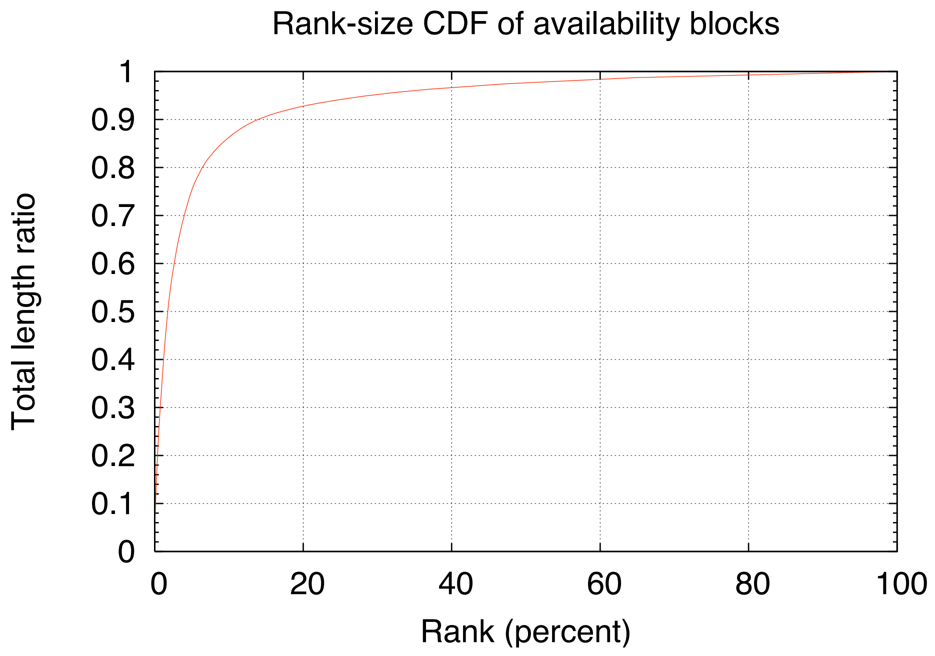

Figure 7.8: Rank-size cumulative distribution function of the length of availability blocks of user 4 of Waterloo dataset.

According to Figure 7.7, 20% of the availability blocks in the dataset contain about 90% of the total availability time [1]. This observation holds for any single user too. For example, figure 7.8 plots the rank-size cumulative distribution of a single user (user 4) in the Waterloo dataset.

By detecting the largest availability blocks, which are also the easiest to detect, the scanning strategy is able to use about 90% of the total available wireless opportunity. This explains the relatively low missed opportunity of both Static Scanning and Exponential Back-off strategies on both datasets (see the average row of Tables 7.2 and 7.4)

The Static Scanning strategy performs fewer scans compared to Exponential Back-off for most of the users (users 2, 3, 4, 5, and 6 in the Waterloo dataset and users 5, 7, 8, 9, and 10 in the Rice dataset). All these users have relatively higher number of opportunity blocks compared to the other set of users. For each disconnection, the Exponential Back-off strategy starts aggressive scanning until, as a result of the back-off mechanism, the scanning interval increases. When the number of disconnections is relatively high (as it is for the listed users), the Exponential Back-off strategy ends up performing too many scans. This can be reduced by making d_0 larger, but this is not studied further in this thesis

[1]In other words, a few "elephants" contain most of the mass of the population.

For some users EB performs fewer scans with high missed opportunity

For user 1 in the Waterloo dataset and users 1, 2, 3, 4 and 8 in the Rice dataset, the Exponential Back-off strategy performs fewer scans and misses most of the wireless opportunity. This is again a result of the number of availability blocks of these users. All these users have relatively few availability blocks, therefore their unavailability time blocks are relatively large. This causes the Exponential Back-off strategy to enter a regime of long scanning intervals, therefore it ends the day with very few scans. As a consequence of long intervals the Exponential Back-off strategy misses many of the opportunities.

Therefore the statement in Chapter 4, that unbounded Exponential Back-off strategy enters very large scanning interval regime only if the availability rate is low appears to be false for power-law distributed unavailability blocks. In fact the number of the availability blocks matters too.

Conclusions

As a result of the power-law distribution of the wireless availability times for mobile users, the Exponential Back-off strategy presents two undesirable behaviors. If the number of availability blocks in a day is relatively high, it performs too many scans and if this number is relatively low it misses too much opportunity. The combination of these two observations makes the Exponential Back-off strategy a bad choice as a scanning strategy.

On the other hand, the Static Scanning strategy performs well both in terms of the number of scans and the missed opportunity. The low variance of the number of scans in Tables 7.1 and 7.3 proves that the Static strategy with 5 minute scanning intervals consistently performs about 200 scans per day and misses about 10% of the opportunities.

7.3 The Optimal Scanning Interval

The scanning interval of the Static strategy is a configurable parameter that affects its performance metrics: the number of scans, and the missed opportunity. This section analyzes the effect of varying the length of the scanning interval to find an optimal value.

Figures 7.9 and 7.10 plot the number of scans (i.e. energy consumed) and the missed opportunity vs. the scanning interval for each user in the Waterloo and Rice datasets, respectively. The number of scans performed during a day is inversely proportional to the scanning interval, and sharply drops as the scanning interval increases. However, the ratio of missed opportunity to the maximum achievable opportunity increases linearly for most users. The slopes of the fitted lines to the

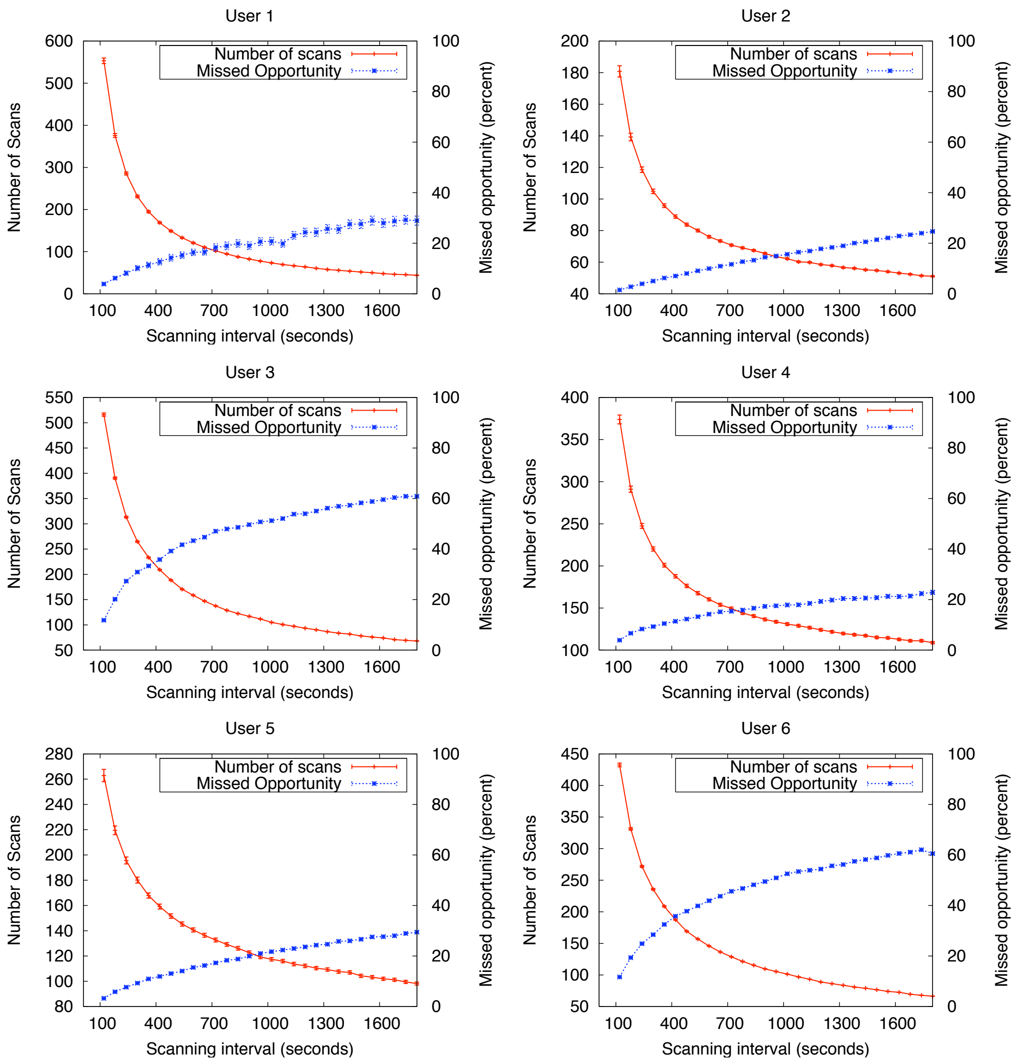

Figure 7.9: The number of scans and percentage of missed opportunity vs. scanning interval of the Static strategy for all the users of the Waterloo dataset.

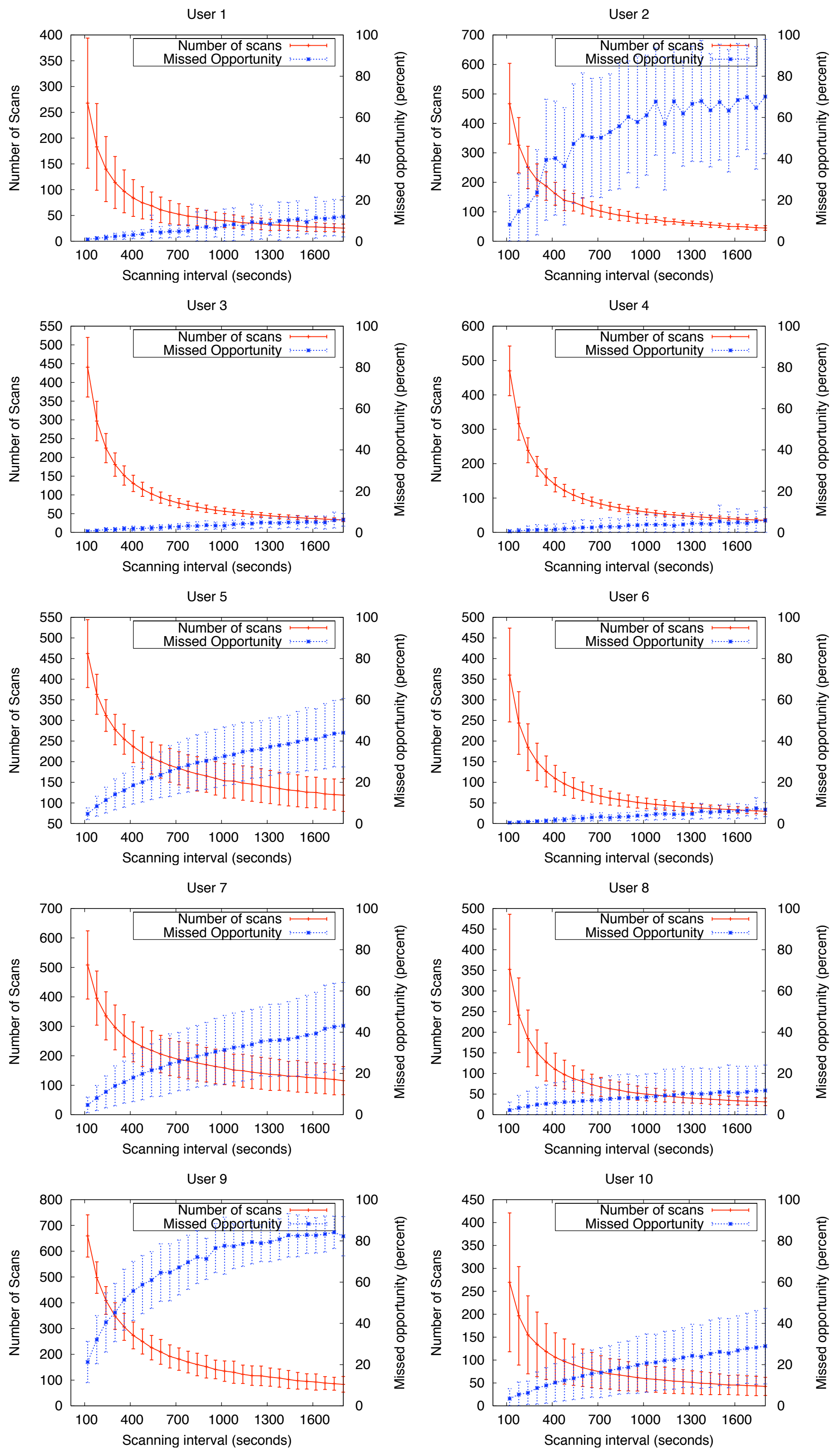

Figure 7.10: The number of scans and percentage of missed opportunity vs. scanning interval of the Static strategy for all the users of the Rice dataset.

user	Slope	Asymptotic Error
Waterloo 1	0.014	±0.000
Waterloo 2	0.013	±0.000
Waterloo 3	0.023	±0.001
Waterloo 4	0.009	±0.000
Waterloo 5	0.014	±0.000
Waterloo 6	0.024	±0.001
Rice 1	0.006	±0.000
Rice 2	0.030	±0.003
Rice 3	0.003	±0.000
Rice 4	0.003	±0.000
Rice 5	0.020	±0.000
Rice 6	0.004	±0.000
Rice 7	0.020	±0.000
Rice 8	0.005	±0.000
Rice 9	0.029	±0.002
Rice 10	0.014	±0.000

Table 7.5: The slopes of the fitted lines to the missed opportunity vs. scanning interval plots for both dataset users, and the asymptotic standard errors.

missed opportunity and their Asymptotic Standard Error are summarized in Table 7.5

The relatively low slope of the missed opportunity vs. scanning interval line, and the sharp decline in the number of scans per day with increased scanning intervals suggest that for delay-tolerant applications, larger scanning intervals are more desirable. If the background applications are truly delay-tolerant, gaining an extra 10% of the connectivity opportunity in a day does not make a big difference, whereas avoiding about 300 scans (see user 4 in Figure 7.9) does make a big difference in the battery life.

However, for applications that do have delay or bandwidth requirements, I present a simple technique that, given the connectivity requirement of the user, can automatically adjust the scanning interval of the Static strategy to minimize the number of scans, and satisfy the user requirement.

This technique is based on three observations: First, that the variance of the missed opportunity ratio of Waterloo users, in Figure 7.9, is relatively low. This is not true for most of the Rice users, probably because of the relatively high number of missing samples in the Rice dataset. To confirm this, note that users 4, and 8 of the Rice dataset, who have relatively few missing samples (see Table 5.4), have tight error bars in Figure 7.10.

The second important observation is that for most users the ratio of the missed opportunity is a linear function of the scanning interval. This claim is proved by the very low standard error values of the linear fitted lines in Table 7.5. And the third observation is that the variance of the availability rate for each user is also very low (see Tables 5.2 and 5.4)

Based on these observations, using the following technique, the smartphone can learn the slope of the missed opportunity vs. scanning interval in at least two days. Knowing this value, the NIC scheduling algorithm can adjust the scanning interval of the Static Scanning strategy based on the user's requirement on the maximum missed opportunity per day. For example, for user 2 of the Waterloo dataset, if the user is not willing to miss more than 10% of the WiFi opportunities of each day, the scanning interval should be set to 10 minutes.

In the learning phase, the smartphone uses at least two different scanning intervals , d_1, and d_2, and uses each in a separate day. d_1 should be relatively small and d_2 should be relatively large. The scheduling algorithm on the smartphone measures the total connected time in each day, T_1, and T_2. The availability rate, $1 - \lambda$, can be estimated as equal to T_1. With at least two points, the missed opportunity ratio line can be estimated.

7.4 Summary

In this Chapter I presented the results of the several simulations with OCMS on both the Waterloo and the Rice datasets. The experiments with different types of schedulers suggest that the Exponential Back-off strategy either performs too many scans, compared to Static Scanning, or misses too many connectivity opportunities. The second problem can be fixed by bounding the maximum back-off time of the algorithm. However, due to its initial aggressive scanning after each disconnection, it will always perform more scans compared to the Static strategy with a carefully selected scanning interval.

In fact, the simulation results suggest that the Static Scanning strategy always detects a high fraction of the total wireless availability time in a day. This phenomenon is a result of the power-law distribution of the length of the wireless availability blocks of wireless mobile users. When there is such a distribution, a few "elephants" contain most of the mass of the population. These elephants are very easy to capture and the Static Scanning strategy will detect them nearly independently of the length of the scanning interval. The length of the scanning interval affects how much of the other non-significant availability blocks can be used by the smartphone, and because their total size is not considerable, varying the interval does not have a significant effect on the missed opportunity. However, increasing the scanning interval dramatically decreases the number of scans that the interface performs per day.

These observations suggest that, for background and delay-tolerant applications Static Scanning with relatively large scanning intervals should be used. I also presented a method that can be used to find the best scanning interval based on the user's expectation of the WLAN NIC scheduling algorithm on maximum missed opportunity time.

Chapter 8

General WLAN Interface Management

Based on the results of the previous chapters, this chapter introduces a generalized scheme for managing WLAN interfaces on smartphones. I consider two distinct classes of applications, based on their connectivity requirements. A *background process* is an application or process running on the smartphone without considerable direct interaction with the mobile user. These applications, in general, are tolerant to delay and disruption of communications. Examples of such applications include file sharing programs, or even an opportunistic email application that tries to upload/download emails with large attachments over WiFi.

Applications that have direct interaction with the mobile user, *interactive processes*, are sensitive to delay and connectivity disruption. For example, when the user is using the web browser, he/she needs network connectivity immediately. In this chapter, I study the effect of such applications on the proposed WLAN interface management strategy.

Section 8.1 reiterates and formalizes the Static Scanning strategy and the proposed technique for scanning interval selection. It also presents an important improvement technique that can reduce the number of scans without increasing the missed opportunity ratio of any scanning strategy. In Section 8.2 I add the user's interaction to the picture and measure the additional cost and evaluate the overall performance.

8.1 Background Processes

The simulations and discussions in Chapter 7 concluded that the Static Scanning strategy with relatively long scanning intervals is the most effective approach to managing the WLAN interface for delay-tolerant applications. Also for a typical mobile user, if gaining an extra 10% of a day's wireless connectivity opportunity does not affect the correct operation of the delay-tolerant applications, then the longer scanning intervals are better.

If the delay-tolerant processes' data requirements cannot be satisfied by relatively long scanning intervals (i.e. too many WiFi opportunities are missed everyday) the WLAN scheduler can be instructed with the maximum acceptable missed opportunity and it can adapt the scanning interval accordingly.

Here, I present a caching technique to reduce the number scans needed by any scanning strategy. Such techniques can be useful if the application requirements make the scheduler's scanning interval too short, and thus incur too much energy cost as a result of aggressive scanning.

8.1.1 Caching Scan Results

Cellular interfaces [1] on smartphones constantly exchange messages with neighboring cell towers. Thus, a smart phone has, at all times, a list of IDs of nearby cell towers This list of currently visible cell towers can be used as a freely-available location hint. The proposed caching technique exploits these location hints to reduce the number of WLAN scans. In this work I used GSM smartphones, and therefore work with GSM cell tower IDs.

In this approach, every time a WLAN scan is performed, the set of detected access points is stored along with the list of visible cell tower IDs in a cache. Before performing a scan, the cache is queried using the visible cell IDs, and if it does not return an answer, the scan is performed and the result is cached. If the cache does return an answer, which consists of a set of BSSIDs, the interface will try to actively associate with them. If none of the access points exists or is usable, another scan will take place and its result will replace the current entry in the cache.

Figures 8.1 and 8.2 summarize the hit ratio of this simple caching technique on the Waterloo and Rice datasets, respectively. In these figures, hit ratio is the ratio of the number of the times that the cache returns an answer to the total number of times it is queried. The answer returned by the cache may not be correct. The false negative ratio is the ratio of the times when the cache falsely returns an empty set to the total number of times it returns an answer. False positive ratio is the ratio of the number of times the BSSIDs returned by the cache are not actually present.

For some of the users this cache is seen to be performing well: user 1 in the Waterloo dataset, and users 3, 4, 6, and 10 in the Rice dataset. The overall performance of this simple cache is satisfactory as summarized in Table 8.1.

These results are from simulations over only one day. In other words, the contents of the cache at the end of each day is not used or the next day. Using the cache from previous day or days will definitely improve the hit ratio. Also, I have been using simple set equality to match two sets of GSM IDs. There are a number

[1]In this work GSM interfaces

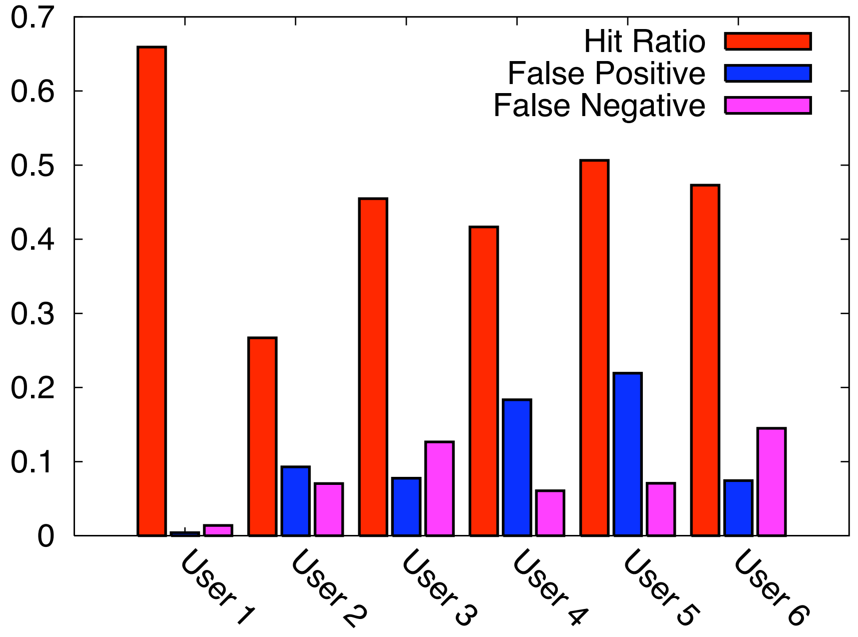

Figure 8.1: Hit ratio, false positive and false negative ratios of the proposed caching scheme on Waterloo users.

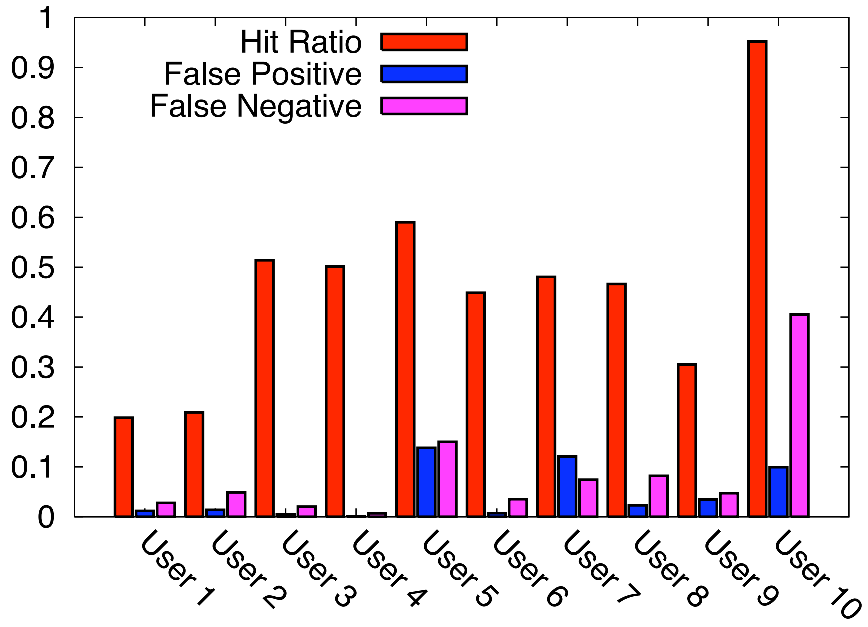

Figure 8.2: Hit ratio, false positive and false negative ratios of the proposed caching scheme on Rice users.

Dataset	**Hit ratio**	**False Positive**	**False Negative**
Waterloo	0.46	0.25	0.19
Rice	0.46	0.16	0.25

Table 8.1: Average hit ratio, false positive, and false negative of the proposed caching scheme on the Waterloo and Rice datasets. False positive and false negative values are conditioned to a cache hit.

of better ways to compare two sets in this case, such as approximate equality, and fuzzy equality. I anticipate that the false positive cases will diminish, if more sophisticated methods are used for matching the query with the cache entries.

8.2 Interactive Processes

If an interactive processes needs network connectivity it is likely that the communication is delay-sensitive. Therefore, if there is a wireless opportunity at that time and in that location, it should be used. If the WLAN interface is already connected to an access point, the interactive process will use the connection, but if the interface is disconnected, the user would probably not like to wait until the next scan, especially if the scanning interval is long.

It seems a natural strategy for the WLAN interface to start a scan as soon as the interactive process starts. This will guarantee that all possible wireless opportunities are exploited for the delay sensitive applications. However, this might have an adverse effect on the number of scans that the interface performs at the end of the day. To study the effects of such a strategy on the number of scans and the missed opportunity, I simulated the combination of background scanning and user-initiated scans on the Waterloo wireless dataset using the user interaction data collected in my measurements.

I modified the Static Scanning strategy to respond to user interactions with an immediate scan if the interface is not connected. I used two weeks of the first user interaction subject and three weeks of the second subject. These five weeks of user interaction data were used in conjunction with the five weeks of each of the Waterloo wireless dataset users. I assume every time the user is working with the keyboard, an interactive process needs network connectivity. This is obviously an overestimation for the interactive processes, and results in a worst-case scenario.

Figure 8.3 plots the average number of scans performed by the Static strategy, using 300 seconds intervals, with and without interactive processes. Figure 8.4 is the same plot for scanning intervals of 900 seconds. These figures show that the number of scans performed by this new scheduler is not significantly higher than the number of scans that the Static strategy normally performs. Also the difference between these two values seems to be larger when the scanning interval increases. This is confirmed in Figure 8.5. This figure plots the number of scans performed

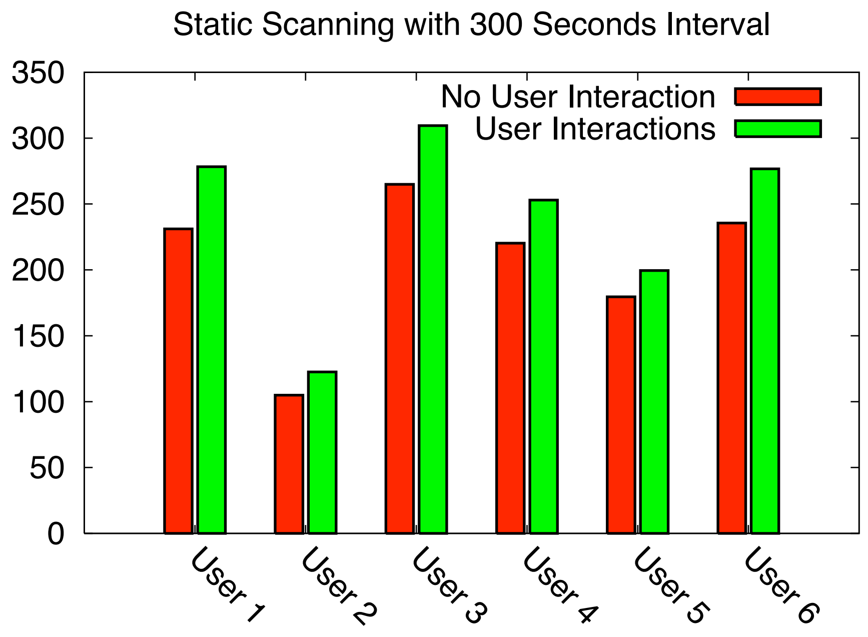

Figure 8.3: Number of scans of the Static Scanning strategy with scanning interval of 300 seconds, with and without the presence of interactive processes.

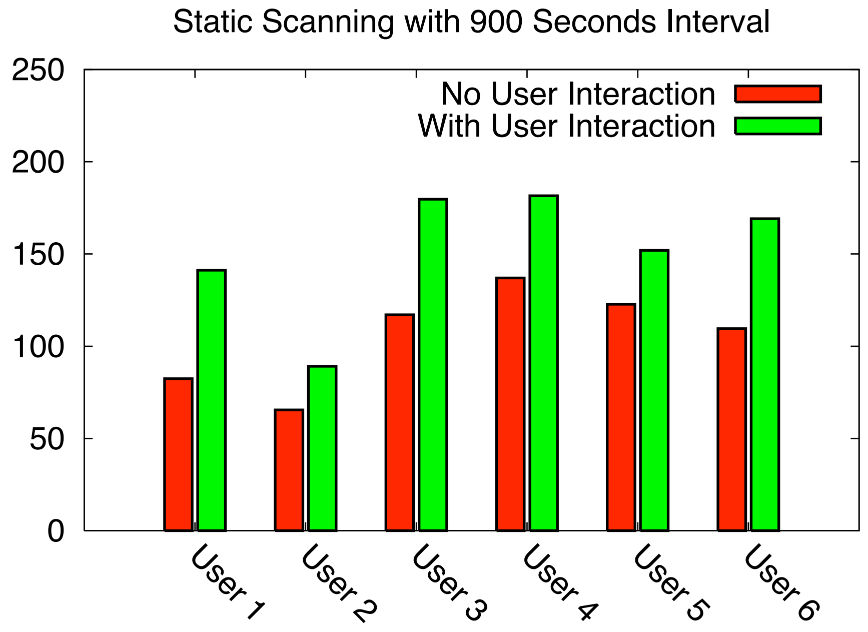

Figure 8.4: Number of scans of the Static Scanning strategy with scanning interval of 900 seconds, with and without the presence of interactive processes.

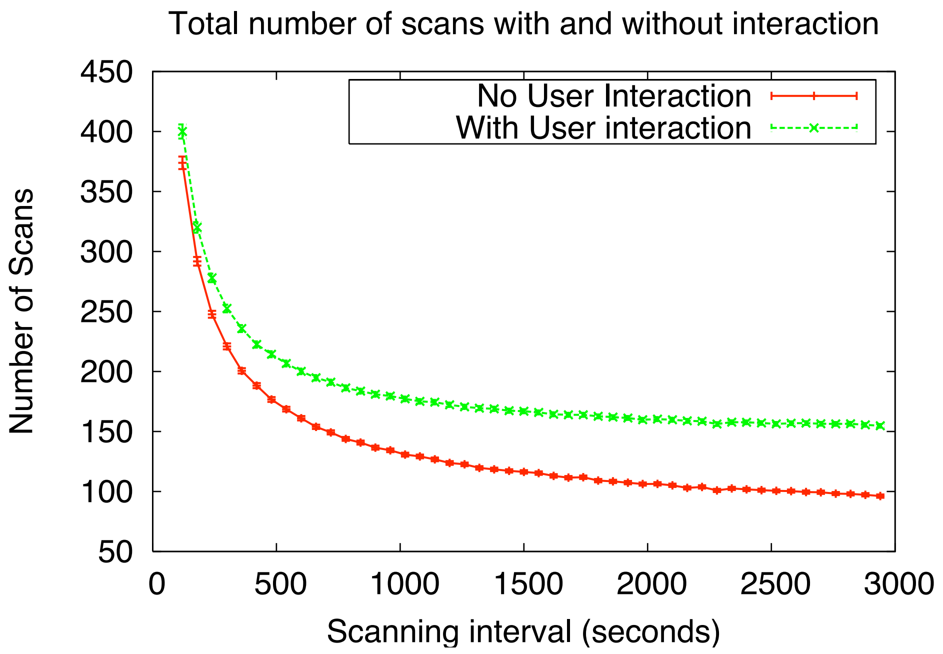

Figure 8.5: Number of scans of the Static Scanning strategy with scanning interval of 900 seconds, with and without the presence of interactive processes.

with and without user interaction scans for a typical user of the Waterloo dataset. The difference between the two cases grows as the scanning interval increases. In fact when the scheduler reacts to user interactions, the number of scans does not drop below a threshold (about 160 in this case).

The extra connectivity time captured by these extra scans is not significant, as expected from the low slope of missed opportunity vs. scanning interval plots in Chapter 7. Table 8.2 summarizes the effect of reacting to interactive processes in terms of scans and the connectivity time. Overall, the additional scans caused by user interactions do not have a significant effect on the performance of the static scanning. Although the extra 1% connectivity is insignificant, it can considerably increase user satisfaction.

I attribute the relatively low impact of user-initiated scans, on the overall performance of the Static Scanning to the high performance of this strategy. The Static strategy already captures most of the wireless connectivity opportunities, therefore many of the user interactions take place at times when the interface is connected, and cause no extra scans.

User	Background Scans	Total Scans	Percentage	Extra Connectivity
1	231	278	20%	1.48%
2	104	122	17%	0.41%
3	264	309	17%	2.52%
4	220	252	26%	0.12%
5	179	199	11%	0.63%
6	235	276	17%	2.05%
Average	106 ± 51	239 ± 62	16 ± 0.02	1.20 ± 0.88

Table 8.2: Summary of the number of scans with and without considering interactive processes, and the extra connectivity time. The scanning interval of the Static strategy is 300 seconds. Background scans are the scans performed by the Static strategy when there is no user interaction.

8.3 Summary

In this chapter I proposed and evaluated two improvements to the Static Scanning strategy for WLAN interface management. First, using context, such as location, hints to reduce the number of scans performed by the interface. A simple way of doing so is taking advantage of the already available GSM cell tower IDs. I showed that a cache of the previous scans that is indexed by the visible cell tower IDs, even with a naive indexing metric, can have average hit ratio of 46%, and the results of the cache queries are relatively accurate.

Second, I augmented the Static Scanning strategy to address delay-sensitive applications. If the interface performs a scan immediately every time the user starts interacting with the smartphone, about 20% extra scans will be performed at the end of the day. For a typical user, most of the user interactions take place at times when the interface is already connected, and only about 20% of them cause unnecessary scans.

Chapter 9

Conclusions and Future Work

This chapter concludes the thesis and outlines possible future work. I also enumerate my recommendations for mobile system designers to manage the WLAN interface on smartphones.

9.1 Conclusions

I investigated WLAN interface management strategies on power constrained mobile devices, such as smartphones. The design properties of the 802.11 standard require that, to achieve maximum power saving, the interface should be turned off if it is not being used (i.e., there is no wireless opportunity). However, the interface should be up and scanning to discover wireless opportunities. Any strategy that manages the WLAN interface should be aware of this delicate trade-off. In this thesis I investigated this trade-off theoretically, statistically and based on real-world data.

If the scheduler has future knowledge, I proved that a dynamic programming algorithm always produces the optimal strategy. In some cases, a simpler greedy algorithm can also result in the optimal solution.

I proposed four practical heuristic scanning strategies and evaluated their energy cost and missed opportunity with simple assumptions on the input. The Naïve strategy incurs too much energy cost. The natural remedy to this problem leads to the Static Scanning strategy that delays scans by a fixed time. I showed how scanning interval can be used to control the operation of the Static strategy with respect to the energy-opportunity trade-off. A more sophisticated strategy for scanning is Exponential Back-off that adapts its scanning interval by doubling it every time a scan fails. If the wireless opportunities arrival rate is from an exponential distribution, and the rate of wireless opportunity in a day is high enough, this strategy achieves a good balance in the trade-off. I proposed the Bounded Exponential Back-off strategy for the cases where the wireless availability rate is low.

To verify the analysis results, I used user traces from two measurement studies, one of which done specifically for this work, and found that the input follows a power-law distribution. Simulation of the proposed strategies on the user traces showed that the Exponential Back-off strategy's aggressive initial scanning incurs a high energy cost, while the Static strategy can capture most of the wireless opportunity time with fewer scans. As a result of the long tail distribution of the length of availability time blocks, a few very long blocks contain most of the availability time of a user's typical day. These long blocks are very easy to capture by the Static strategy, even when the scanning interval is relatively large.

I studied the trade-off between the energy consumption and missed opportunity of the Static strategy with respect to the length of the scanning interval. The number of scans is inversely proportional with the length of the scanning interval, while the missed opportunity increases almost linearly with the length of scanning interval, with a small slope. Therefore, the longer scanning intervals should be preferred. Based on the linear relation between missed opportunity and scanning interval, I proposed a simple technique to adapt the number of scans according to the minimum missed opportunity requirement of mobile delay-tolerant applications.

To address delay-sensitive applications that typically interact with the mobile users, I proposed combining the Static Scanning strategy with a reactive strategy that performs a WLAN scan every time the user turns on the mobile screen. I evaluated this new strategy using the wireless traces and the user interaction traces that I collected as part of my experiment. The number of extra scans that result from the user interactions is not statistically significant.

I also studied the performance of a simple caching technique that uses GSM cell tower IDs as indexes into a cache of previous scan results. The average hit ratio of the cache is 46% and the number of false negative and false positive results is not significant. Therefore, I recommend using similar measures to decrease the number of scans on the smartphone.

9.1.1 Recommendations

Based on the findings in this thesis I make the following recommendations for mobile system designers and software developers.

1. Provide mechanisms for mobile applications to state their delay sensitivity to the mobile Operating System or connectivity management software layers.

2. For delay-tolerant applications use Static Scanning with the largest possible scanning interval. The largest possible interval can be found using the technique outlined in Chapter 7 of this thesis.

3. For delay-sensitive applications use an aggressive scanning mechanism. The over-all penalty of the most aggressive scanning strategy for interactive applications is shown to be insignificant.

4. Using context hints can help avoiding unnecessary scans. A simple cache of the previous scans that uses cell tower IDs can avoid about 50% of the scans with relatively low false results.

9.2 Future Work

This thesis can be extended and continued in the following directions.

1. **Considering Usability of Access Points:** In this work I assumed the access points that are not WEP enabled are usable. I eliminated what appears to be sporadically seen access points through high-pass filtering. But there is still some concern about the usability of frequently visited access points. The work done by Nicholson *et al.* [25] can be extended to this study for more accurate results. My hypothesis is that eliminating unusable access points will not affect the power-law distribution of the availability blocks.

2. **Improvement of Caching:** My investigations with the caching scheme serve as a proof of concept. The performance of such a cache can potentially be improved by taking advantage of more sophisticated set matching or learning algorithms.

3. **Smarter Interactive Scans:** I used the most aggressive scanning strategy for interactive and delay-sensitive processes. This strategy does not appear to affect the performance of background scans. However, the user interactions and other context hints from the mobile user can be used in smarter ways. Furthermore, investigation of users' interactions with their smartphones can open the possibility of better scanning strategies both for delay-tolerant and delay-sensitive applications.

4. **Management of Multiple NICs:** The relatively high wireless availability rate and the power-law distribution of wireless opportunity times can be used for better management of multiple network interfaces on smartphones. This information may be useful in reducing the complexity of the interface selection problem. The information received from other network interfaces can also help in management of the WLAN interface. A simple example of this is the caching technique that is introduced in this thesis.

5. **Collaborative Scheduling:** Multiple smartphones in the same space can collaborate with each other to identify usable access points. It is also possible for smartphones to selectively share their past experience (e.g., the WLAN scan cache) with other smartphones. The effectiveness of such techniques can be investigated using the wireless traces collected in this work.

References

[1] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 179–191, 2007. x, 1, 2, 5

[2] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and transparent dynamic content updates for mobile clients. *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 56–68, 2006. 1, 4, 14

[3] D. Ashbrook and T. Starner. Learning significant locations and predicting user movement with GPS. *Wearable Computers, 2002.(ISWC 2002). Proceedings. Sixth International Symposium on*, pages 101–108, 2002. 7, 8

[4] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003. 7

[5] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular internet access using in situ Wi-Fi networks. *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 50–61, 2006. 6, 14

[6] H. Cai. Crossing over the bounded domain: from exponential to power-law inter-meeting time in manet. *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 159–170, 2007. 6

[7] A.T. Campbell, S.B. Eisenman, N.D. Lane, E. Miluzzo, and R.A. Peterson. People-centric urban sensing. *Proceedings of the 2nd annual international workshop on Wireless internet*, 2006. 4

[8] A. Clauset, C.R. Shalizi, and MEJ Newman. Power-law distributions in empirical data. *Arxiv preprint arXiv:0706.1062*, 2007. 26, 30, 33

[9] M. Denny, M. Franklin, P. Castro, and A. Purakayastha. Mobiscope: A Scalable Spatial Discovery Service for Mobile Network Resources. *Proc. of the 4th International Conference on Mobile Data Management (MDM)*, pages 307–324. 4

[10] K. Fall. A delay-tolerant network architecture for challenged internets. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003. 4

[11] R. Hariharan and K. Toyama. Project Lachesis: Parsing and Modeling Location Histories. *Geographic Information Science*, 2004. 7, 8

[12] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001. 7

[13] J.H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(3):58–68, 2005. 7, 8

[14] Srinivasan Keshav. Cell phones as a research platform. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 2–2, New York, NY, USA, 2007. ACM. 40

[15] D. Kotz and T. Henderson. Crawdad: A Community Resource for Archiving Wireless Data at Dartmouth. *IEEE PERVASIVE COMPUTING*, pages 12–14, 2005. 5, 22

[16] B. Kreller, D. Carrega, J. Shankar, P. Salmon, S. Bottger, and T. Kassing. A Mobile Aware City Guide Application. *Proc. of ACTS Mobile Communications Summit, Rhodes, Greece*, pages 60–65, 1998. 7

[17] K. Laasonen, M. Raento, and H. Toivonen. Adaptive On-Device Location Recognition. *Proc. Pervasive*, 4, 2004. 7

[18] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. *Proceedings of Pervasive*, 3468:116–133, 2005. 7

[19] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft. Opportunistic content distribution in an urban setting. *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 205–212, 2006. 6

[20] A. Lindgren, C. Diot, and J. Scott. Impact of communication infrastructure on forwarding in pocket switched networks. *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 261–268, 2006. 6

[21] N. Marmasse and C. Schmandt. Location-Aware Information Delivery with ComMotion. *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, pages 157–171, 2000. 7

[22] M. McNett and G.M. Voelker. Access and mobility of wireless PDA users. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(2):40–55, 2005. 6

[23] M. Motani, V. Srinivasan, and P.S. Nuggehalli. PeopleNet: engineering a wireless virtual social network. *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 243–257, 2005. 4

[24] A. Nicholson and B. Noble. BreadCrumbs: Forecasting Mobile Connectivity. *Proceedings of the 14th annual ACM international conference on Mobile computing and networking*, 2008. 8, 13

[25] Anthony J. Nicholson, Yatin Chawathe, Mike Y. Chen, Brian D. Noble, and David Wetherall. Improved access point selection. In *MobiSys*, pages 233–245, 2006. 1, 6, 71

[26] D.J. Patterson, L. Liao, D. Fox, and H. Kautz. Inferring High-Level Behavior from Low-Level Sensors. *International Conference on Ubiquitous Computing (UbiComp)*, 2003. 7

[27] T. Pering, Y. Agarwal, R. Gupta, and R. Want. CoolSpots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232, 2006. 5

[28] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, Jon Crowcroft, and Christophe Diot. Experiments in mobile social networking. Technical Report CR-PRL-2008-02-0003, Thomson, February 2008. 4

[29] LR Rabiner. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 8

[30] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 165–178, 2007. 5, 20, 22

[31] I. Ramani and S. Savage. SyncScan: practical fast handoff for 802.11 infrastructure networks. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 1, 2005. 4

[32] N. Ravi, J. Scott, L. Han, and L. Iftode. Context-aware Battery Management for Mobile Phones. *Microsoft Research Tech Report.* 5, 7

[33] S. Shin, A.G. Forte, A.S. Rawat, and H. Schulzrinne. Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs. *Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 19–26, 2004. 19

[34] W. Wang, V. Srinivasan, and M. Motani. Adaptive contact probing mechanisms for delay tolerant applications. *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 230–241, 2007. 4

[35] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992. 7

[36] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999. 2

[37] M.A. Zaharia and S. Keshav. Fast and Optimal Scheduling Over Multiple Network Interfaces . Technical report, Technical Report cs-2007-36, University of Waterloo, 2007. 5

[38] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen. Discovering personal gazetteers: an interactive clustering approach. *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 266–273, 2004. 7