# Collaborative and adaptive mobile device-resident service architectures

Emmanouil Koukoumidis

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Electrical Engineering
Advisers: Margaret R. Martonosi and Li-Shiuan Peh

January 2012

# Abstract

Mobile devices, such as smartphones and personal media players, have recently significantly increased in popularity thanks to the rich set of mobile cloud services that they allow users to access. Networked vehicular computing devices are also expected to be commonplace in the near future, as they will enable a wide range of driver assistance services. The ubiquitous penetration of mobile services, however, has been thwarted by their poor user experience; access to mobile cloud services typically occurs over slow and costly long-range cellular communications.

This thesis focuses on improving the user experience of mobile services by reducing the need for costly long-range cellular communications. To achieve this, the thesis proposes to host more service functionality on mobile devices themselves. In this way, mobile devices are often able to serve requests either locally or by contacting neighbor devices over short-range communications. Two novel mobile service architectures are proposed for the two different types of mobile services: traditional non-geo-locality services and emerging geo-locality services.

A service is termed to have the geo-locality property when its data are both generated (sensed or input) and consumed locally, *i.e.,* within a specific geographic region. In other words, for services that have the geo-locality property, only mobile devices within a specific geographic region $R$ can generate the necessary service data and only devices within the very same region $R$ are interested in consuming it. For non-geo-locality services, the data is generated either by cloud servers or by users regardless of their location. Data generation and/or consumption are also typically a function of the users' personal interests and not of their geographic location.

For traditional non-geo-locality services, this thesis proposes the Pocket Cloudlets architecture. The Pocket Cloudlets architecture is a mobile device-resident caching scheme that serves cloud service requests locally on the device, when possible, significantly reducing the need for slow and costly long-range communications. The Pocket

Cloudlets architecture leverages both personal user and collaboratively-generated community access patterns to selectively replicate parts of the cloud service locally on the mobile device. Pocket Cloudlets are also adaptively updated by detecting emerging popular service data items and prefetching them on the mobile device. Our analysis shows that the proposed Pocket Cloudlets architecture can effectively augment several traditional cloud services, like mobile web search. PocketSearch, our prototyped mobile search pocket cloudlet, reduces the average service access time by a factor of 2.7× and the required communication bandwidth by 66%.

For emerging geo-locality services, the thesis presents the Region-Resident Services (RegReS) middleware. RegReS allows a rich set of emerging geo-locality services to be fully supported on confederations of mobile devices. Mobile devices collaborate to provide a geo-locality service within a specified region and over a specified service lifetime by utilizing only short-range ad-hoc communications. In this way, RegReS completely eliminates the need for long-range cellular communications. Although mobile devices are becoming increasingly powerful, their resources are constrained and should be used judiciously. RegReS enables the efficient provision of geo-locality services by allowing services to specify their target service carrier density. Only as many service carriers as specified are subsequently maintained by RegReS. As opposed to previously proposed static schemes, RegReS employs a fully distributed, collaborative and adaptive estimation scheme to track the existing service carrier density and make decisions about the spawning of new carriers, when necessary. Thanks to collaboration and adaptation, RegReS can maintain the desired density with only 16% mean absolute error across a wide range of configurations.

To demonstrate the potential of collaborative mobile device-based computing platforms that are enabled by middleware like RegReS, the thesis presents a rich set of novel services that such platforms can enable. More specifically, the thesis focuses on the type of services that are typically most challenging and resource-intensive (*e.g.,* CPU), camera-based services. We introduce five such services and prototype SignalGuru, a camera-based traffic signal schedule advisory service. SignalGuru leverages opportunistic sensing and collaboration across windshield-mounted smartphones and their cameras to provide drivers with information about the schedule of traffic signals ahead. Results from two deployments of SignalGuru, using iPhones in cars in Cambridge (MA, USA) and Singapore, show that traffic signal schedule can be predicted with very good accuracy. On average, SignalGuru comes within $0.66s$, for pre-timed traffic signals and within $2.45s$, for traffic-adaptive traffic signals. Feeding SignalGuru's predicted traffic schedule to our Green Light Optimal Speed Advisory (GLOSA) application, our vehicle fuel consumption measurements show savings of 20.3%, on average. SignalGuru information can also be fed into several other envisioned applications to further improve fuel efficiency, vehicle flow, travel time and road safety. The example of SignalGuru illustrates that with collaboration and adaptation, mobile device-based computing platforms can support a rich set of challenging services without the help of cloud servers and the associated long-range communications.

Overall, this thesis advocates and demonstrates that, with collaboration and adaptation, mobile devices can effectively support a rich set of services and thus reduce the need for slow and costly long-range cellular communications to cloud servers. Several traditional cloud services that operate on very large amounts of data can be selectively and adaptively hosted on mobile devices. Furthermore, novel mobile services that may seem prohibitively resource-intensive and challenging can be enabled and hosted on confederations of collaborating mobile devices. In this way, the mobile user experience can be greatly improved and a significant amount of the increasingly scarce long-range communication bandwidth can be saved.

# Acknowledgements

If PhD were to be described as a journey, then it would have to be called an Odyssey. A long, tough journey. Often, the sea is rough, your ship gets wrecked and you need to figure out how to make your own raft in order to set sail again towards your destination. But then again, there will often be no wind in your sails and you will have to row, row and row... However, at the same time such a journey teaches you a whole lot on both a technical and personal basis. When coupled with two influential advisors, this trip becomes a life-changing experience that instills a diverse set of skills and values. Friends and family are also there to provide their support and make this journey more enjoyable.

First and foremost, I would like to thank my two advisors Li-Shiuan Peh and Margaret Martonosi for making my PhD such a rich and rewarding experience. Margaret Martonosi always supported my summer expeditions into industry and Li-Shiuan Peh was instrumental in making my academic studies interstate and intercontinental, from Princeton to Boston, from Boston to Singapore and all the way back. At the same time, Margaret and Li-Shiuan instilled into me their technical approach and work ethic preparing me to become a successful professional. Margaret has always been coaching me to improve my non-technical skills as well. Furthermore, Li-Shiuan's optimism and excitement has always served as an example for maintaining a positive attitude.

I would also like to express my gratitude to the rest of my thesis committee and thesis research collaborators. Mung Chiang, Niraj Jha and Jennifer Rexford provided valuable feedback to improve the content and presentation of this thesis. Dimitrios Lymberopoulos and Jie Liu offered me a very enriching and exciting internship at Microsoft Research.

Furthermore, I truly value the eager help of Sarah McGovern, Maria Rebelo and Michelle Ho with all the administrative issues. Lisa Schreyer and Olivia Martel made the Graduate Student Government Events Board meetings a lot more fun to chair.

An invaluable part of my PhD, were the friends that I met in Princeton, Boston and Singapore. Thanks to them, the five years of my PhD passed a lot more enjoyably. With Konstantinos Aisopos, we set off on our PhD journey together and travelled together to Princeton, MIT and Singapore. We missed Greece and mourned about it together but we also laughed a lot. I would also like to thank Yu-Yuan Chen and Carole-Jean Wu for all the fun we had together and for hosting me, as I write this thesis, in 5Y-Magie. Geogios Vasilakis, thank you for being such a great friend, always eager to help out. Thank you Ilias Tagkopoulos, Eli Berici and Silvana Endler for the great dinners that we had together. Alex Ntelekos, thank you for the two-dollar-note tradition that you initiated. As the third possessor of the lucky note, I will make sure I pass it on to a trusted person. Najwa Aaraj, thank you for being such a great teaching assistant and a great friend ever since. Jakub Szefer, thank you for encouraging me to go swimming more often and for all the great times that we had in Princeton and California. Abhishek Bhattacharjee, Niket Agarwal, Arun Raman, Prateek Mishra, Tushar Krishna and Anirudh Sivaraman, thank you for all the fun times we had together, the volleyball that we played and the Hindi that you taught me. I have particularly used "Kali Ma Shakti De" at numerous tough occasions during this long PhD journey. Speaking of useful phrases, thank you Chia Hsin (Owen) Chen for reminding about "hakuna matata." Finally, I would like to thank Ozlem Bilgir, Gilberto Contreras, Eric Chi, James Donald, Jason Gao, Chrysanthos Gounaris, Sibren Isaacman, Wenhao Jia, Constantinos Kalfarentzos, Ioannis Kamitsos, Kevin Ko, Amit Kumar, Vincent Lenders, Caterina Liviakis, Dan Lustig, Chris Sadler, Vassos Soteriou, Eleftherios Tsiokos, Yavuz Yetim, Huayong Wang and Pei Zhang for making Princeton, Boston and Singapore a better place to be.

Last but not least, I would like to thank my father, my mother and my fiancée, Maria Vasilaki, for their support throughout my long PhD Odyssey. My father was the one that initiated me into electrical engineering and has always served as my primary motivating example of a skilled technical person. My mother has always been encouraging me and reassuring me that I would be arriving at my Ithaca soon. After five years, I can now discern Ithaca far in the distance. It now seems that she will most likely prove right. Maria was a constant source of support and the most pleasant resort for my thoughts during both good and bad times. As a small sign of gratitude, I would like to devote this thesis to them.

To my father, mother and Maria.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction: Rise of Networked Mobile Devices

Networked mobile devices are becoming increasingly pervasive. The number of mobile Internet users is expected to far exceed the number of desktop Internet users and reach 1.8 billion by the end of 2014 [101]. Smartphones are popular because they allow their mobile users to access many different types of information through a rich set of cloud services. Mobile users can use a search engine service to locate and access any type of information that resides within the World Wide Web while on-the-go. Users can also access real-time information pertinent to their commute (*e.g.,* traffic information, bus schedules, *etc.*) as well as several other types of location-based information via dedicated cloud service applications and interfaces. Mobile devices are expected to soon become the dominant computational devices and the major means of access to the Internet and other sources of information.

Besides personal media players and smartphones, networked vehicular computing devices are expected to be commonplace in the near future as well. Already, several vehicles (*e.g.,* Lexus IS) are equipped with specialized onboard computational units that have not only location (*e.g.,* GPS) and motion sensors (*e.g.,* accelerometer) but also wireless interfaces like Bluetooth. In addition, vehicular computational devices are expected to be augmented with more powerful short-range communication interfaces (*e.g.,* 802.11p [61, 148]). These new wireless interfaces will effectively enable vehicle-to-vehicle and vehicle-to-infrastructure communications for the purpose of supporting novel driver-assistance applications.

Mobile devices are not only increasing in numbers but also becoming increasingly computationally powerful and sensor-rich. This is particularly the case for smartphones. Dual-core smartphones clocked at 1.2GHz are already in the market and quad-core ones are expected soon. Smartphones are also typically equipped with microphone, GPS, accelerometer, gyroscope, compass, ambient light sensor, one or more cameras, touch screen and multiple communication interfaces (cellular, Wi-Fi, Bluetooth). Communication interfaces can be used not only for communication but also for sensing RF signals. Near Field Communication (NFC), radar, temperature, humidity and gas sensors may be commonplace in future smartphones as well. In Section 1.5, the technology trends for mobile devices are presented in greater detail.

Thanks to their pervasiveness, increased computational capabilities and rich set of sensors, smartphones have become the main means for mobile users to contribute real-time data. Users share their comments and reviews about events in real-time by publishing comments in their preferred online portal [154] or social network [35] and augmenting them with photos that they capture with their phone's cameras. Users can also decide to contribute their GPS and accelerometer data as they are drive or ride the bus. By crowd-sourcing such information from everyday users, an accurate and comprehensive real-time traffic updates cloud service can be supported [99]. Information crowd-sourced from mobile devices can also be used to detect road abnormalities [33], collect information for available parking spots [19, 98], track the location of transit vehicles [144] and measure air or noise pollution [116].

A big collaborative ecosystem of mobile users is thus emerging. In contrast to traditional desktop users, mobile users are not only consumers but also producers of many different types of real-time data. These data comes either from manual user input or from automatic sensor readings and can be used to improve current services or enable new ones altogether.

## 1.2 Motivation: Shortcomings of Existing Mobile Cloud Service Architecture

Commercial mobile services are typically hosted on Internet cloud servers accessed over long-range communications. Cloud servers aggregate, process and maintain all the service data. Mobile devices need to contact the cloud server whenever they want to consume data or contribute data that they have generated (input by user or sensed by device sensors). Mobile devices, despite their increased capabilities, are used as dumb terminal devices just for input/output. For example, in the case of a parking spot availability service [98], all data about detected parking spots are uploaded to a cloud server that is potentially located hundreds or thousand of miles away. Users looking for parking spots will then have to contact this server to learn about availabilities.

The traditional cloud service architecture is ill-suited for mobile services. The cloud service architecture was originally designed for desktop users with highly available and fast wired network connectivity. Cloud services, however, are typically accessed by mobile devices over costly long-range cellular Internet communications. Long-range communications typically provide good coverage in residential areas but come with several side effects. These side effects sometimes make their use impractical or lead to poor quality of service.

First, cloud services when accessed on mobile devices suffer from high network latencies. Unlike the desktop domain where the connection to any cloud service

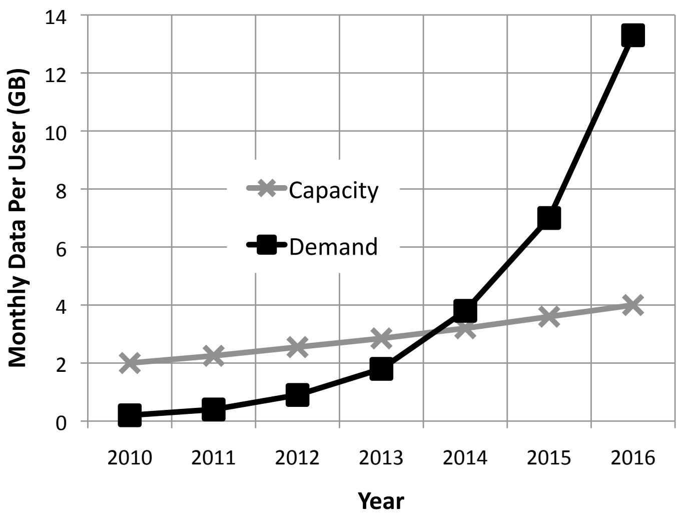


Figure 1.1: **Cellular bandwidth demand versus capacity [127].**

takes place over very fast and almost always available links (i.e., Ethernet), in the mobile domain users rely on cellular radios that tend to exhibit significantly higher latency and unpredictability. For instance, the response time for a web search service is typically at least one order of magnitude higher on a 3G-connected smartphone compared to an Ethernet-connected desktop machine [58, 81].

Second, the cellular bandwidth, on which the existing mobile cloud service architecture depends, is a scarce resource. At peak times, some cell towers are already getting overloaded. The cellular data bandwidth is expected to become soon an even scarcer resource. As Figure 1.1 shows, the average monthly demand for cellular data bandwidth is expected to have exceeded the available capacity by 2014. In 2016, the demand is expected to be three times greater than the possible capacity despite the integration of new technologies (*e.g.,* 4G). The architecture for mobile services needs to change. Scarce long-range cellular bandwidth should be used only when necessary.

Third, long-range cellular radios impose not only a latency and bandwidth, but also a power bottleneck. Mobile devices are battery-operated and the cellular radio, along with the processor and the screen, is one of the most power-hungry components. The more data are exchanged and the more time the radio link is active, the lower the battery lifetime of the mobile device becomes, creating a negative user experience. Short-range communications are significantly more power-efficient (Section 2.6.1).

Fourth, long-range cellular radios are not always available. Several types of mobile devices like laptops, Internet tablets and personal media players (*e.g.,* iPod) are typically equipped only for short-range communications. Long-range communications come at an additional cost of purchasing higher-end models or extra hardware. As a result, the use of long-range communication is sometimes not even an option. Mobile services should be re-architected and made accessible over short-range communications, when possible.

Last but not least, even when long-range cellular communications are available, their use for Internet access comes with significant monetary costs to users. The inclusion of an unlimited data plan can significantly increase the cost of a cell phone contract. As a result, users may be unwilling to undertake that cost and instead, opt for limited data plans. In such plans, cellular providers charge users a relatively high premium for each KB of data that they upload or download. Moreover, roaming charges for data services are often prohibitive when traveling. Therefore, cellular Internet communications should be used judiciously.

## 1.3 Proposed Approach: Collaborative and Adaptive Mobile Device-resident Services

Depending heavily on long-range communications, the traditional mobile cloud service architecture leads to poor experience for mobile users. In order to alleviate the shortcomings of the traditional cloud service architecture, this thesis proposes to host mobile services either partially or even fully on the mobile devices themselves. In this way, requests are served locally on the device or over short-range communications, when possible, and long-range communications are avoided. To enable mobile device-resident services, the thesis introduces two alternative service architectures for the two different types of mobile services; traditional non-geo-locality services and emerging geo-locality services. A service is termed to have the *geo-locality property* when its data is both generated (input or sensed) and consumed locally, *i.e.,* within a defined geographic region. The differences between non-geo-locality and geo-locality services are summarized in Table 1.1. In Section 1.6, the properties of these two types of services and the proposed architectures for them are discussed in more detail. To be effective, both the proposed mobile service architectures and the services running atop employ inter-device collaboration and adaptation to the variable environment parameters. As our results show, collaboration and adaptation are both critical.

## 1.4 Related Work: Caching of Services

To avoid poor wireless communications and improve performance, both commercial and proposed academic approaches employ caching. Besides mobile services, temporal and spatial locality in data accesses has been widely leveraged to selectively cache parts of the data and thus minimize the need for slow network or next-level memory accesses. Caching has been employed in a wide variety of systems ranging from processor chips [48, 136], to operating systems [11, 146], network file systems [6, 105], database systems [13, 41, 135], distributed mobile device-based services [19, 95], cloud-based services [15, 96, 112, 115] and others. In these works, several different cache-replacement policies have been proposed. Some proposed caching schemes also employ intelligent prefetching [14, 52, 113, 147] methods to further improve the cache hit rate.

### 1.4.1 Non-Geo-locality Cloud-based Services

For traditional non-geo-locality cloud services, caching has been extensively employed to reduce latency and server load. Caching has been commercially deployed or proposed

Table 1.1: **Key differences between non-geo-locality and geo-locality services.**

| | | **Non-geo-locality Services** | **Geo-locality Services** |
|---|---|---|---|
| Service Properties | **Data Type** | Typically very large in size; data of global importance. | Relatively smaller in size; data of only local importance. |
| | **Data Generation** | Typically location-independent. Generated by cloud server or geographically distributed users as a function of their personal interests. | Location-dependent. Input/sensed by users/devices within a specific geographic region. |
| | **Data Consumption** | Typically location-independent. Based only on user's personal interests. | Location-dependent. Consumed by users within a specific geographic region (same region where data was initially generated). |
| Proposed Service Architecture | **Challenge** | Very large data set; need to filter with caching and prefetching for power/performance. | Constrained mobile device resources for service and data hosting; need to trade off with service access latency and sensing coverage. |
| | **Data Caching** | Location-independent. Depends only on personal user interests. | Location-dependent. The type/subset of data (sensed and) cached varies depending on location. |
| | **Request Serving** | From local cache when possible. Otherwise from cloud server over long-range communication. | From local cache when possible. Otherwise from neighboring mobile devices over short-range communication. |

to improve access to distributed network file [6, 105] and database systems [13, 41, 135], DNS [112], web content [15, 97, 113, 115] and search results [96]. Deployed and proposed approaches for cloud service caching are based on a multi-tier architecture in which the data are cached both on the cloud server and on intermediate proxy devices. Several of these services, including web content, are heavily cached on the user's mobile device as well. For others (*e.g.,* web search), however, only cloud-based (server, proxy) caching schemes have been proposed.

For mobile cloud services, in particular, local caching on the user's mobile device is the most critical. As explained in Section 1.2, mobile cloud services suffer from the last mile problem. The long-range communication to cellular base stations is the bottleneck in mobile cloud service access that leads to poor user experience. As a result, alleviating the need for long-range communications to proxy devices or the actual cloud servers yields by far the most performance benefits.

This thesis advocates that several popular services that were traditionally thought to belong to the cloud, can be effectively cached on mobile devices as well. To enable such caching, the thesis proposes the Pocket Cloudlets architecture, a mobile device-resident caching architecture. The design of the proposed architecture is based on technology trends that this thesis analyses. Five services that can be supported on the Pocket Cloudlets architecture are briefly described and a mobile search engine service is analyzed and evaluated in detail.

### 1.4.2 Geo-locality Mobile Device-based Services

The idea of location-based services, *i.e.,* services that are of interest and should be provided only to nodes within a specific geographic region, started as early as 1987, when Finn proposed location-based routing [37]. The interest in location-based services increased significantly after 1994, when GPS became fully operational. Ever since node location information has been widely used for node addressing and routing

[37, 74, 104], service replication and placement [52, 54], sensor activation [54, 86], to establish trust in user-generated content [90] and other purposes. Furthermore, an increasing number of location-based services have been proposed. Location-based services recently proposed include traffic advisory [99], road conditions advisory [33], parking spots availability information [19, 98], air and noise pollution information [116].

Although all these proposed services possess the geo-locality property, most of them [33, 98, 99, 116] depend on cloud servers to cache and maintain the sensed information. However, as discussed in Section 1.2, this traditional cloud-based mobile service architecture has significant limitations.

In contrast, some other proposed approaches leverage the mobile devices themselves to cache and maintain the geo-locality service's information across time. Two main mobile device-based service-maintenance approaches have been proposed for unreliable and uncontrolled mobile networks: 1) push-based schemes [19, 89, 92, 93, 95] that push and assign the service (data caching and sensing) on all nodes within a region and 2) pull/subscription-based schemes [91, 95] in which only nodes interested in using (pulling) the service will maintain (cache and sense data for) it.

None of these two schemes can support geo-locality services effectively. Push-based schemes are wasteful; in dense urban environments that most of these proposed services target, it is often redundant and wasteful to use all the available mobile devices. Mobile device resources are constrained and should be used judiciously. At the same time, pull-based schemes may lead to poor service quality; often a disparity will exist between the number of the nodes that are interested in consuming the service (*e.g.,* parking spot availabilities) and the number of nodes that are necessary to support it efficiently (detect most available parking spots robustly). A critical capacity of service carriers is necessary for each geo-locality service. While using more service carriers is wasteful, using fewer may lead to poor service quality.

To enable efficient maintenance of geo-locality services, this thesis proposes the Region-Resident Services (RegReS) middleware. In contrast to previous work, RegReS allows services to specify their desired service carrier density, *i.e.,* the number of mobile devices per unit area that should be providing the service (caching and sensing of data). RegReS maintains this density in a fully-distributed, collaborative and adaptive approach, attacking key challenges of emerging everyday user mobile device networks (*e.g.,* smartphone and vehicular networks) without the need for cloud servers and long-range communications to reach them.

## 1.5 Trends in Mobile Device Technology

To demonstrate the ever-increasing potential of mobile devices, in general, and smartphones, in particular, as a powerful computing platform for the hosting of mobile services, this section presents, in more detail, mobile device technology trends.

### 1.5.1 Compute

The computational power of smartphones is already significantly high (6000 DMIPS and 1GB RAM for Samsung Galaxy S II) and is continuing to increase at an exponential rate. For more than two decades, computer systems have followed the exponential performance improvement trends predicted by Moore's Law [100, 107]. As shown in Figure 1.2, smartphone application processors have followed similar trends from 2004 to 2009. In 2011, after smartphone application processors reached the 1GHz clock frequency, technology scaling started being leveraged to build dual-core processors (Figure 1.3). Chip multi-processors have the potential to deliver higher performance per watt compared to a more complex single processor of the same size [108]. By 2013, quad-core smartphone processors are expected as well. Beyond 2013, the compute power of computer systems, in general, and smartphones, in particular, will continue

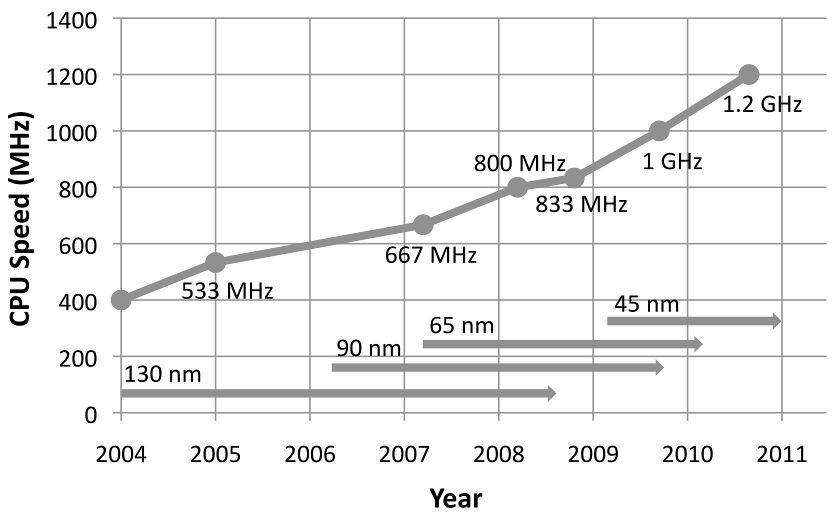


Figure 1.2: **Samsung ARM roadmap for mobile device application processors [128]. ARM processors dominate the mobile and embedded electronics market.**

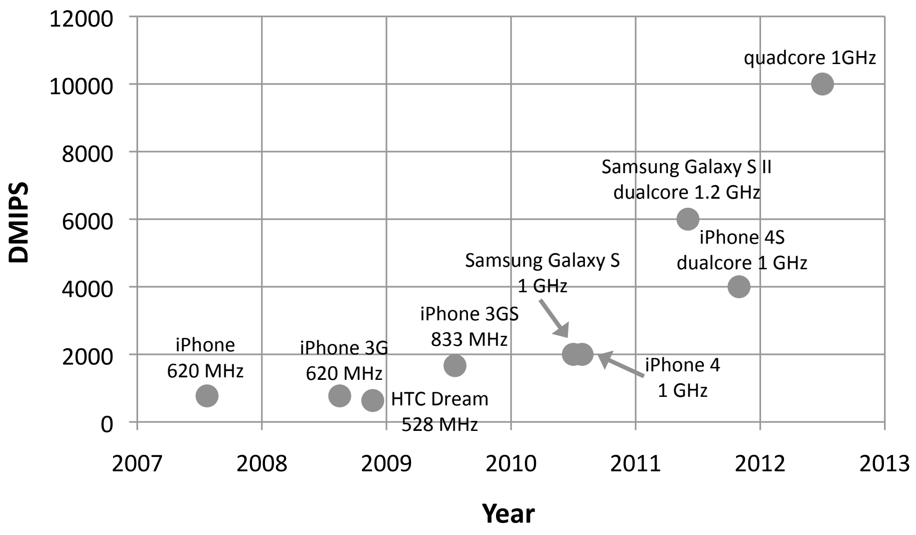


Figure 1.3: **Samsung ARM processor performance trend for smartphones from 2009 to 2013 [2, 3]. Processor performance is measured in Dhrystone [152] Million Instructions Per Second (DMIPS).**

to grow fast but possibly at a slower rate [34] primarily because of power dissipation problems [8, 9, 124]. RAM has also been following similar exponential scaling trends.

Smartphones will have substantial compute capabilities, which are not leveraged effectively by the current cloud service-based computing model. More intelligence can

Table 1.2: **Technology scaling trends.**

| | **Flash** | | | | **Other NVM technology** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **year** | ’10 | ’12 | ’14 | ’16 | ’18 | ’20 | ’22 | ’24 | ’26 |
| **tech (nm)** | 32 | 22 | 16 | 11 | 11 | 8 | 5 | 5 | 5 |
| **scaling factor** | 1 | 2 | 4 | 8 | 8 | 16 | 32 | 32 | 32 |
| **chip stack** | 4 | 4 | 6 | 6 | 8 | 8 | 12 | 12 | 16 |
| **cell layers** | 1 | 1 | 1 | 2 | 2 | 4 | 4 | 8 | 8 |
| **bits per cell** | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |

be pushed to the mobile devices so that the typical access to cloud servers over poor long-range communications is avoided, when possible.

### 1.5.2 Storage

In order to be able to assess the potential of mobile device-resident cloud services, it becomes critical to know what storage capacities are expected to be available on mobile devices. We thus perform here an extensive analysis of Non Volatile Memory (NVM) trends for the dominant mobile devices, *i.e.,* mobile phones.

Table 1.2 shows our somewhat conservative technology scaling projections in the NVM market through 2026. We assume that flash will dominate this market until it runs into charge-based storage scaling issues in the 2016/2018 time frame. At this point, we assume it will be replaced with another NVM technology more resilient to smaller feature sizes, such as resistive (*e.g.*, PCM [117], RRAM [67]) or magneto-resistive memories (*e.g.*, STT-MRAM [57]).

The first data row of Table 1.2 shows a projection of how the number of cells per layer in NVM memory devices is expected to scale in the form of a scaling factor. During the period in which flash is used as the NVM technology of choice, it is projected to double in capacity every two years [40].

In 2018, flash may lose traction due to increasing challenges in storing state (*i.e.,* electrons) in charge-based cells. A new technology providing more stable cells at smaller features may at that point replace flash. Single-layer PCM is already being productized as replacement for NOR flash in mobile devices, so it is a good candidate. The shift from flash to another technology could cause significant disruption in fabrication processes and would likely cause scaling to stall for one generation. Scaling is likely to resume in subsequent years until it finally stops in 2022 when industry is expected to hit the 5$nm$ technology.

The second row of Table 1.2 shows chip stacking projections. Two layers are added every four years until 2022, when the technology may be mature enough to start making increments of four layers every four years.

The third row shows a progression of number of layers when cell stacking is employed. Cell stacking is a technique by which devices are fabricated in multiple layers on the same silicon base (instead of in independently fabricated chips that are then combined, as in chip stacking) [76]. The process of taking a technology from industrial lab prototype to initial production is typically five years[1]. Given that this technology was demonstrated in 2009, this technology is likely to be adopted circa 2016 and the number of layers is likely to double every four years.

Finally, the fourth row of Table 1.2 shows the number of bits per memory cell. This number should increase in the next few years for flash, but then start to decrease as feature sizes get smaller, process variation increases and the average number of electrons per cell drops. In such a setting, even small electron losses may cause cell state to be corrupted, which in turn forces designers to reduce the number of logic levels, and therefore bits per cell, to increase the distance between these levels and better distinguish stored values.

Assuming the trends described above, Figure 1.4 presents various evolution scenarios for NVM parts used in smartphones. Our projections start with the NVM storage

[1]According to Leibson's Law, disruptive technologies take 10 years to become pervasive in the design community. However, our focus here is not on pervasive, but on initial production for high-end mobile device models. Furthermore, chip stacking [76], in particular, has already been demonstrated industrially on a significant scale.

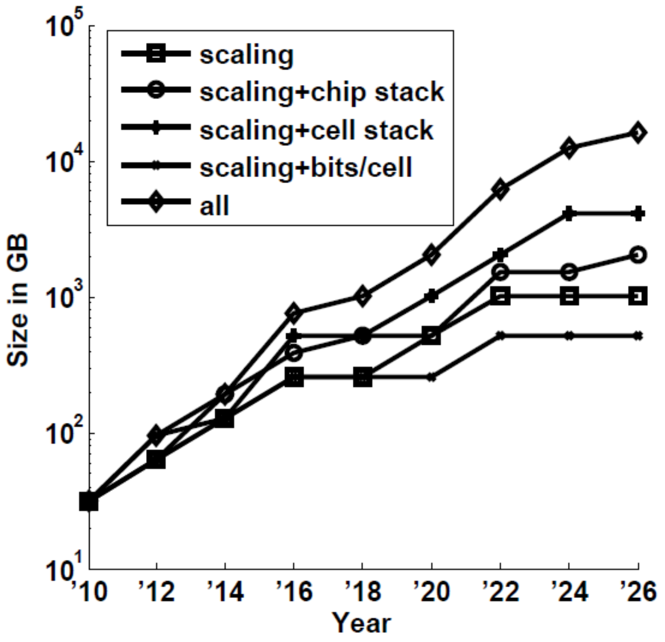


Figure 1.4: **Memory size evolution for high-end smartphones assuming the trends shown in Table 1.2.**

found in a high-end smartphone in 2010. We then apply different combinations of scaling and other capacity-increasing techniques to make a projection of total NVM capacity in future smartphones. Figure 1.4 shows that high-end phones may reach 1 TB of NVM as early as 2018. Considering that low-end smartphones today have 512 MB of NVM, a ratio of 64-to-1 when compared to high-end smartphones, we can calculate that low-end phones may eventually reach 256 GB (16 GB in 2018), still a respectable amount of storage.

As our analysis shows, the already abundant storage capacity of smartphones is expected to continue increasing rapidly. Smartphones are thus expected to become increasingly capable of caching mobile services and associated data in their NVM.

### 1.5.3 Sensors

Mobile devices, in general, and smartphones, in particular, are equipped with an ever richer set of sensors.

**Location Sensors** Smartphones are equipped with sensors that enable the device to localize itself and thus provide location-based services. Before GPS became a pervasive smartphone sensor, smartphones relied on cell tower and Wi-Fi base station triangulation to localize themselves. The localization accuracy with such technologies, however, is low and has very high variance. The accuracy is a few hundred meters for cell tower triangulation [134] and a few tens of meters for Wi-Fi base station triangulation [24, 134].

However, over the last couple of years, GPS has become the most pervasive smartphone sensor offering significantly higher accuracy. Since 2000, when selective availability (intentional satellite signal degradation) was turned off for commercial use, GPS's accuracy improved from 100m down to 10m. The Russian GLONASS [46] and the Chinese BeiDou (COMPASS) [10] global navigation satellite systems have similar performance to the US GPS. With Galileo [43], the European counterpart of these systems, the accuracy will be further improved and reduced down to within 1m. For a fee, Galileo will be able to offer centimeter-level accuracy for civilian purposes as well. Galileo will start becoming operational in 2012 when at least four satellites will be in orbit. Galileo's coverage will continue to improve till 2018 when 30 satellites will have been launched in total.

While GPS offers accuracy that is adequate for most applications, it also has several shortcomings. First GPS [73] has limited availability. GPS requires line of sight with at least four satellites in order to acquire a location lock. As a result, GPS cannot work indoors or even in urban canyons. Second, GPS is a major energy hog for mobile devices [31]. When GPS is not available or when power conservation is necessary, localization based on wireless signal triangulation is used instead in commercial systems. Researchers have also proposed several other alternative approaches for indoor positioning based on acoustic fingerprints [141], geo-magnetism [25] and others. Such approaches can robustly (92% [141]) determine in which room users are located or their actual location within 1-2 meters accuracy [25]. RFID-based localization has also the potential to provide localization with sub-meter accuracy [130] but requires a dense deployment of RFID tags.

Lately, some high-end smartphones (*e.g.,* iPhone 3GS) are equipped with a compass sensor as well. Furthermore, altimeter (barometer sensors) may be added soon to smartphones for measuring elevation. Elevation information could be used, for example, to determine on which floor of a building the user currently is.

**Inertial Sensors** In 2007, Apple equipped the iPhone with an accelerometer to switch its display automatically from portrait to landscape orientation. Ever since, the 3-axis accelerometer has been used to enable a rich set of applications ranging from games to transit tracking [144]. More recently, gyroscopes that can determine the orientation of the device (yaw, roll and pitch angles) have started to be integrated into smartphones as well. When measurements from these two sensors are fused, the mobile device can more robustly and accurately detect changes in the device movement and orientation.

**Audio Sensors** Smartphones typically have one or more microphones. Besides the original goal of enabling phone calls, smartphone microphones have been used to estimate traffic conditions (detecting and localizing honking) [99], mobile phone pairing [114] and for other purposes.

**Visual Sensors** Smartphones are equipped with powerful digital cameras. A back-facing camera is a standard feature for all smartphones enabling the capture of photos and videos. Recently, some smartphones (*e.g.,* iPhone 4) started featuring a front-facing camera as well, in order to enable video calls.

The quality and capabilities of smartphone cameras have been increasing fast; the size of and amount of pixels in digital smartphone cameras are strongly linked to

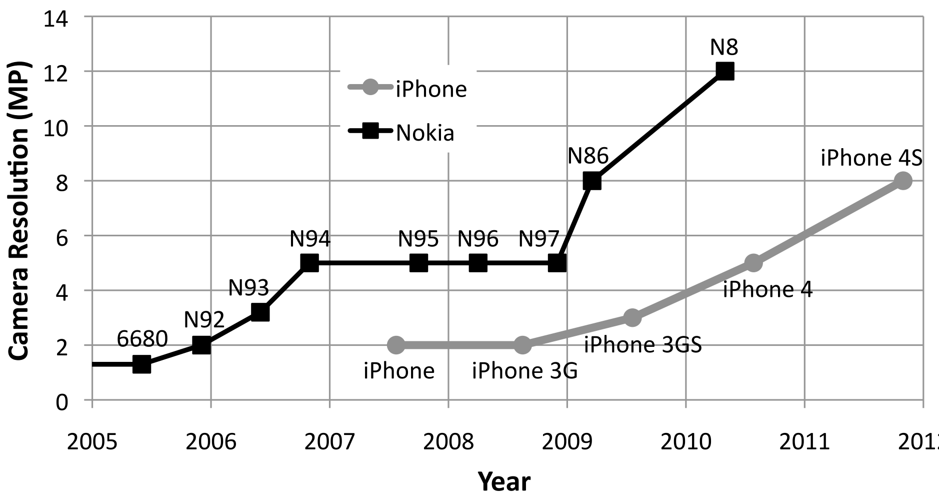


Figure 1.5: **Camera resolution roadmap for iPhone and Nokia smartphones. The resolution of smartphone cameras has been increasing fast.**

Moore's Law. As shown in Figure 1.5, the camera resolution for iPhone and Nokia devices has been increasing at an exponential rate. Nokia has been traditionally featuring the most high-end cameras for its smartphones. At the same time, Samsung has already announced CMOS camera sensors with resolution as high as 16MP [129].

The rate of improvement for CMOS cameras is expected to slow down though, unless new camera technologies are invented [22]. Because of mobile device size constraints, the size of the image sensor has to be relatively small. Therefore, the camera analysis is typically improved by leveraging Dennard transistor scaling and making pixels smaller. In order to decrease the size of the pixel proportionately to Dennard scaling and fit more of them in a given sensor area, not only the transistors but the photosensitive area of the pixel needs to get decreased proportionately as well. However, the smaller the photosensitive area of a pixel, the lower its ability to capture light and thus the lower the quality of the image. There is hence a tradeoff between image resolution and sensitivity. This tradeoff limits the benefits of Dennard scaling. Nevertheless, smartphone cameras are already powerful enough to support most popular applications.

**Other Existing Sensors** Some smartphones have an ambient light sensor to adjust display brightness and thus save battery power. Furthermore, proximity sensors have been introduced into smartphones as well. In the iPhone, the proximity sensor deactivates parts of the hardware (*e.g.,* touch screen and display) when the device is placed next to the user's head in order to conserve battery power and prevent accidental inputs. So far, these sensors have not been widely leveraged to enable other novel functionality and services. In 2010, Nokia introduced a radar sensor [106] into one of its smartphones that can detect objects ahead and report their speed and direction of movement. The radar sensor has not become popular either yet.

**Potential Future Sensors** As mentioned earlier, future smartphones may feature barometer sensors. Temperature, humidity, gas and other environmental sensors may also be integrated in smartphones as well. Furthermore, pervasive smartphones have recently started being used as healthcare assistants. As a result, healthcare-related sensors (*e.g.,* heart monitor, respiratory sensor) may also become commonplace in the future. Several other types of sensors that are currently used only by scientists may eventually be added in commodity smartphones, if a popular application for them is conceived. In short, smartphones will be equipped with an increasing set of sophisticated sensors that can enable new services. Especially, when multiple smartphones collaborate, the aggregate sensing capacity and service provision potential begin to look extremely promising.

### 1.5.4 Wireless Communication Interfaces

There is a very rich variety of wireless network technologies. Three major categories of wireless communication interfaces exist for mobile devices though: Wi-Fi (802.11a/b/g/n), cellular (2G, 3G) and Bluetooth (802.15.1). For each category, several different protocols and corresponding interfaces exist.

As Table 1.3 shows, Wi-Fi, cellular and Bluetooth interfaces exhibit very different performance. Furthermore, the performance of these interfaces exhibits very high variance. Wireless communication channels are a shared resource. Therefore, the performance of wireless networks varies significantly depending on the load created by all the users sharing the same channel. Cellular networks, in particular, are often heavily loaded and the performance experienced by individual users is, on average, at least an order of magnitude lower compared to the quoted numbers [58]. As a result, cellular connections often result in poor user experience and Wi-Fi connections are preferred instead, when available. As shown in Section 2.6.1, querying a search engine via 802.11g is 2.4× faster and 2.2× more power-efficient compared to using 3G. Compared to EDGE, 802.11g is 3.8× faster and 3.9× more power-efficient.

The three aforementioned wireless interface categories can enable different modes of communication. Cellular interfaces can be used only for infrastructure mode connections to the Internet through the cell towers. On the contrary, Bluetooth is used for direct (ad hoc) communication between devices. Wi-Fi can be used both in infrastructure and ad hoc mode.

Smartphones and some other high-end mobile devices typically feature all three interface categories: cellular, Wi-Fi and Bluetooth. They do not support all the standards of these three categories though. Smartphones typically support 802.11b/g and recently some models (*e.g.,* iPhone 4S) have started featuring 802.11n as well. Because of the smaller coverage of 3G networks, smartphones feature not only 3G but also older 2G cellular interfaces. Two Bluetooth-enabled devices can always communicate with each other (at the rate of the slower one), as all Bluetooth versions are backwards compatible. In the near future, vehicles will most likely be featuring 802.11p interfaces [61]. 802.11p is a new standard in the Wi-Fi family that will enable high-rate and low-latency short-range vehicle-to-vehicle and vehicle-to-infrastructure communications.

Table 1.3: **Wireless Network Technologies Comparison [17, 29, 32, 39, 59, 60, 61, 62, 125, 126]. For marketing reasons, EDGE and HSPA+ (recent improvement over HSPA) have been commercially advertised as 3G and 4G networks, respectively. However, according to the International Telecommunication Union specifications [63, 64], EDGE and HSPA+ belong to the 2G and 3G family of protocols, respectively. Cellular networks are configured to have a range of a few hundred meters to a few kilometers depending on population density. Smaller cells allow for better spatial reuse of the wireless spectrum and thus higher aggregate capacity within a defined geographic region.**

| | **Standard** | **Release** | **Range** | **Max. Rate** | **Latency** | **Infrastructure mode** | **Ad hoc mode** |
|---|---|---|---|---|---|---|---|
| **Wi-Fi** | 802.11a | 1999 | 120m | 54Mbit/s | <2ms | √ | √ |
| | 802.11b | 1999 | 140m | 11Mbit/s | <2ms | √ | √ |
| | 802.11g | 2003 | 140m | 54Mbit/s | <2ms | √ | √ |
| | 802.11n | 2009 | 250m | 300-600Mbit/s | <2ms | √ | √ |
| | 802.11p/DSRC | - | 300-1000m | 6-27Mbit/s | 100$\mu$s-50ms | √ | √ |
| **Bluetooth** | 802.15.1 Class 1 | 2002 / 2004 | 100m | 1Mbit/s / 3Mbit/s | <10ms | X | √ |
| | 802.15.1 Class 2 | 2002 / 2004 | 10m | 1Mbit/s / 3Mbit/s | <10ms | X | √ |
| | 802.15.1 Class 3 | 2002 / 2004 | 5m | 1Mbit/s / 3Mbit/s | <10ms | X | √ |
| **Cellular** | 2.5G (GPRS) | 2000 | a few km | 80kbit/s $\downarrow$, 40kbit/s $\uparrow$ | 300-350ms | √ | X |
| | 2.75G (EDGE) | 2003 | a few km | 473.6kbit/s $\downarrow$, 473.6kbit/s $\uparrow$ | <150ms | √ | X |
| | 3.5G (HSPA) | 2007 | a few km | 14Mbit/s $\downarrow$, 5.76Mbit/s $\uparrow$ | 25ms | √ | X |
| | 4G (LTE 2×2 MIMO) | 2009 | a few km | 173Mbit/s $\downarrow$, 58Mbit/s $\uparrow$ | <5ms | √ | X |

802.11p will be operating over the Dedicated Short-Range Communications (DSRC) channels.

The performance of proposed standards for cellular, Wi-Fi and Bluetooth technologies has been improving fast over the last decade. Performance improvements are particularly critical for cellular technologies as cellular networks are already often getting overloaded leading to poor performance. LTE is expected to double the spectral efficiency compared to deployed 3G technologies (*e.g.,* HSPA, EV-DO) by leveraging Multiple-Input Multiple Output (MIMO) technology. However, cellular technologies are reaching the Shannon bound, the theoretical limit for spectral efficiency for a specific signal-to-noise ratio [127]. Directive antennas and smaller cells that allow for better spatial reuse [139] may be leveraged more aggressively in the future to further increase the available capacity.

Despite these technological improvements, the demand for cellular bandwidth is still expected to grow significantly faster than the available capacity. New mobile service architectures are necessary that will ease or completely remove the strong dependence of mobile services on cellular links to Internet servers. Short-range, phone-to-phone communications over WiFi, Bluetooth or the emerging Near Field Communication (NFC) should be leveraged instead to support mobile services.

### 1.5.5 Battery

Battery power is often the most constrained resource for mobile devices. Battery technology has been evolving very slowly compared to other mobile device components, *e.g.,* processor. However, according to Koomey's Law [78], the energy efficiency of computers has been doubling every about one and a half years. This trend has been followed over the last six decades and is expected to continue. This means that at a fixed computing load, the amount of power consumed by a mobile device will be getting reduced by a factor of two about every 18 months. Therefore, despite the small improvements in battery technology, the capabilities of mobile devices will continue to increase at an exponential rate.

## 1.6 Thesis Contributions

Leveraging the ever-increasing capabilities of mobile devices, this thesis proposes novel middleware and architectures to empower mobile devices as the mobile service providers. Two different schemes are proposed and evaluated for the two different types of services: 1) the Pocket Cloudlets architecture [81] for traditional non-geo-locality services and 2) the Region-Resident Services (RegReS) middleware [84] for emerging geo-locality services. The potential of the proposed mobile service architectures is illustrated by describing a rich set of services that each one of them can support. A service is prototyped and evaluated for each of the two proposed architectures. For the Pocket Cloudlets architecture, we prototyped a mobile search service termed PocketSearch. For the RegReS middleware, we prototyped a Parking Availability Service (PAS). In addition, in order to demonstrate the full potential of RegReS-enabled collaborative mobile-device based computing platforms, we also prototyped SignalGuru [85], a novel traffic signal schedule advisory service.

### 1.6.1 Pocket Cloudlets Architecture

For non-geo-locality cloud services that operate on global data (*e.g.,* search engine, *etc.*), this thesis proposes the Pocket Cloudlets architecture [81]. This architecture is a client-side caching scheme that leverages personal user and collaboratively-generated community access patterns to selectively replicate parts of the cloud service locally on the mobile device. Pocket cloudlets are also adaptively updated by detecting emerging popular service data items and prefetching them on the mobile device. Selective replication is critical since the global data (*e.g.,* search index, world wide web, *etc.*)

are normally prohibitively large. The device-resident pocket cloudlet intercepts the requests to the cloud service and provides locally cached results, if possible. In this way, pocket cloudlets limit the use of slow and costly long-range communications.

The Pocket Cloudlets architecture is motivated by mobile device technology trends. Radio and battery technologies will improve over time, but are still expected to be the bottlenecks in future systems. However, as our analysis shows, NVMs may continue experiencing significant and steady improvements in density for at least ten more years (Section 1.5.2). The Pocket Cloudlets architecture leverages the abundance in the memory capacity of mobile devices to mitigate latency and energy issues when accessing cloud services.

The thesis briefly describes five services that could be supported on the Pocket Cloudlets architecture. As a showcase, we present in detail the design, implementation and evaluation of PocketSearch, a web search pocket cloudlet. We perform mobile search characterization to guide the design of PocketSearch and evaluate it with 200 million mobile queries from the search logs of m.bing.com. We show that PocketSearch can serve, on average, 66% of the web search queries submitted by an individual user without having to use the slow 3G link, leading, on average, to a 2.7× service access speedup. The example of PocketSearch demonstrates the impact of the proposed architecture.

Finally, based on our experience with PocketSearch, we provide additional insights and guidelines on how future pocket cloudlets should be organized, from both an architectural and operating system perspective.

### 1.6.2 RegReS: Region-Resident Services Middleware

For geo-locality services, *i.e.,* services that operate on data that are both generated and consumed locally (within a specified region), this thesis proposes the RegReS middleware [84]. This middleware, leveraging the geo-locality property of such services, enables their hosting on confederations of regional mobile devices in a fully infrastructureless fashion. Mobile devices, termed service carriers, collaborate using their short-range communications to maintain the services within a defined geographic region and for a defined service lifetime.

Although mobile devices are becoming increasingly powerful, their resources (computation, memory, storage, communication, battery energy) are still often constrained for certain types of resource-intensive services (*e.g.,* SignalGuru). Mobile devices should thus be used judiciously when supporting such services. Unlike proposed approaches for geo-locality services, RegReS allows services to specify their desired critical density of service carriers. The critical density of carriers is service-specific and ensures that the service neither runs low on capacity nor wastes more resources than necessary. The RegReS middleware, which runs on the mobile devices, ensures that this target carrier density is maintained in a fully-distributed fashion. RegReS uses a collaborative estimation scheme to track the existing carrier density for a service. RegReS then employs spawn policies and carrier selection criteria to decide when and which regional mobile devices to spawn as new service carriers.

Furthermore, unlike other proposed static estimation schemes, RegReS adapts the parameters of its estimation scheme to system dynamics. As our results show, adaptation is critical for maintaining a stable and accurate density of service carriers in highly volatile systems like vehicular ad hoc networks. In such networks, node mobility (*e.g.,* average speed) varies a lot across the day depending on the prevailing traffic and other road conditions.

RegReS is able to maintain a stable and accurate density of service carriers across a wide range of configurations. Results from the ORBIT testbed [111], using synthetic and real bus mobility traces, show that RegReS adapts to different system configurations, preserving the desired service density with less than 16% mean absolute error. By maintaining the requested target carrier density accurately, RegReS can form an important foundation for low- or even zero-infrastructure mobile services.

### 1.6.3 Novel RegReS-enabled Geo-locality Services: SignalGuru

The RegReS middleware can fully support a rich set of proposed services whose data are both sensed and consumed locally. Examples of such geo-locality services include traffic advisory [99], road conditions [33], parking spot availability [19, 98], air or noise pollution [116] services. Our RegReS-based PAS sniffs broadcast reports to learn about the release of parking spots and subsequently provides this information to vehicles in the vicinity that are looking for a free spot.

To demonstrate the full potential of RegReS-enabled computing platforms that are based on collaborating mobile devices, we focus on the challenging and compute-intensive class of camera-based services. To enable grassroots camera-based services, we propose a novel sensing platform that is composed of windshield-mounted smartphones and their cameras. As the vehicles move, smartphones opportunistically capture video frames with their cameras and process them to detect target objects. We envision that many already proposed or new services can be supported by such a grassroots sensing platform. These services include but are not limited to a free parking spot availability service, bus localization and arrival time service, free taxi discovery service, cheap gas advisory service and SignalGuru [85].

Our proposed SignalGuru is a novel traffic signal schedule advisory service. SignalGuru leverages the cameras of windshield-mounted smartphones to detect traffic signals and their current status, merge the locally detected information with received information and ultimately predict the traffic signals future schedule. The traffic signal schedule that SignalGuru predicts can then be fed into several envisioned applications to improve fuel efficiency, vehicle flow and road safety. Our work on SignalGuru makes several other contributions as well. We propose a lightweight iterative traffic signal detection algorithm and a two-stage filter to compensate for this lightweight yet noisy detection scheme. Furthermore, we propose schemes to predict the schedule of both pre-timed and traffic-adaptive traffic signals.

Results from two deployments of SignalGuru, using iPhones in cars in Cambridge (MA, USA) and Singapore, show that traffic signal schedules can be predicted accurately thanks to collaboration and adaptation. On average, SignalGuru comes within $0.66s$ for pre-timed traffic signals and within $2.45s$ for traffic-adaptive traffic signals. Without collaboration the prediction error for traffic-adaptive traffic signals would be $4.5\times$ higher ($11.03s$). Furthermore, frequent enough adaptation (retraining) of the Support Vector Regression (SVR) traffic signal models, reduces the prediction error by 25%. Feeding SignalGuru's predicted traffic schedule to our GLOSA application, our vehicle fuel consumption measurements show savings of 20.3%, on average.

Our work on SignalGuru demonstrates that even challenging camera-based services can be effectively supported on confederations of mobile devices without long-range communications to cloud servers. It also demonstrates that such grassroots services can deliver significant benefits, both monetary and environmental. We believe that the effectiveness of SignalGuru will motivate further research in services that RegReS, in general, and our proposed novel sensing platform, in particular, can enable.

## 1.7 Thesis Outline

The following chapters describe the contributions of this thesis in detail. Chapter 2 presents the Pocket Cloudlets architecture for traditional non-geo-locality services. Chapter 3 describes the design of the RegReS middleware for emerging geo-locality services. The potential of collaborative mobile device-based computing platforms that are enabled by middleware like RegReS are next illustrated in Chapter 4 by describing five geo-locality services that such platforms can support and presenting in detail the design and evaluation of SignalGuru. Finally, Chapter 5 offers the conclusions of this thesis and provides pointers for future research.

# Chapter 2

# Pocket Cloudlets Architecture for Non-geo-locality Services

This chapter presents the Pocket Cloudlets architecture that is targeted at non-geo-locality cloud services and helps reduce their need for long-range communications. The proposed architecture is a cloud service cache architecture that resides on the mobile device's non-volatile memory and, when possible, services requests locally on the device. This architecture utilizes both personal user and collaboratively-generated community access models to maximize its hit rate and, subsequently, reduce overall service latency and energy consumption. The effectiveness of the proposed architecture is explored and demonstrated with PocketSearch, a web search pocket cloudlet. We evaluate the improvement in user experience (bandwidth and latency reductions) that PocketSearch can deliver and the resources (memory, NVM) needed. The applicability of the Pocket Cloudlets architecture for more mobile services is also examined. Finally, this chapter provides additional architectural insights regarding the operation of one or more pocket cloudlets on a mobile device.

## 2.1 Introduction

The availability of Internet access on mobile devices, such as phones and personal media players, has allowed mobile users to access valuable information through a rich set of cloud services. Thanks to search engine services, mobile users can locate and access any type of information that resides within the world wide web. Mobile users can also access many special types of information (*e.g.,* restaurant reviews) via dedicated cloud service interfaces. The ever-increasing set of offered services has spurred a growing interest in accessing these services even while on the go.

However, the traditional cloud service architecture leads to poor user experience for mobile users. In the desktop domain, services are accessed over fast and stable connections. In contrast, in the mobile domain, services are accessed over slow, power-hungry, costly and less predictable long-range cellular communications. For example, while the typical response time for a web search service on a desktop machine is a few hundred milliseconds, on a 3G-connected mobile device, it is at least an order of magnitude higher (3 to 10 seconds depending on location, device and operator used). When the 3G radio is not connected or only EDGE connectivity is available, this response time may be doubled or even tripled. Besides being slow, power-hungry and costly, long-range cellular communications are also becoming an increasingly scarce resource.

On the other hand, storage on mobile devices is becoming increasingly abundant. Phones and personal music players currently support up to 64 GB of flash memory and future flash technologies promise much higher capacities. The increasingly available memory resources on these devices can transform the way mobile cloud services are structured. As more and more information can be stored on mobile devices, specific parts of or even full cloud services could be transferred to the actual mobile devices, transforming them into *pocket cloudlets.*

Pocket cloudlets could drastically improve the mobile user experience in three major ways. First, users can instantly get access to the information they are looking for. When local results are available, the cost, latency and power bottleneck introduced by the cellular radio is eliminated. Furthermore, by serving user requests on the actual device, pocket cloudlets can mitigate pressure on cellular networks, which is expected to be a scarce resource as mobile Internet grows. Second, since most of the interactions between the user and the service take place on the mobile device, it is easier to personalize the service according to the behavior and usage patterns of individual users. Third, since the service resides on the phone, all the personalization information could also be stored on the phone and possibly protect the privacy of individual users.

Pocket cloudlets can thus enable a fast and personalized mobile user experience. The proposed Pocket Cloudlets architecture for non-geo-locality mobile services takes advantage of the increasing Non-Volatile Memory (NVM) sizes to alleviate shortcomings of cellular radios and above all the latency and power bottlenecks. For instance, in the case of a search engine, a large part of the web index, ad index and local business index can be stored on the phone's NVM enabling users to instantly access search results locally without having to use the cloud. In essence, a mini search engine could be running on the phone providing real-time search results to the mobile user. In another setting, the actual web content could also be cached on the mobile device to provide an instant browsing experience. Web content that might be of interest to the user could be automatically downloaded to the user's phone overnight and become available during the course of the day.

The structure of this chapter is as follows. First, Section 2.2 briefly describes different mobile services that could be potentially architected as pocket cloudlets. Then Section 2.3 proposes the pocket cloudlet architecture that leverages both personal user and community models to selectively cache cloud service data in mobile device

NVM. In Section 2.4, we analyze 200 million queries to understand how mobile users search on their phones, and then utilize the results of this analysis to guide the design of PocketSearch (Section 2.5), a pocket cloudlet that replicates a search and advertisement engine on an actual phone. Next in Section 2.6, using a prototype implementation and real search query streams extracted from mobile search logs of `m.bing.com`, we show that PocketSearch is able to successfully serve, on average, 66% of all web search queries submitted by an individual user. The benefits are twofold. From the user perspective, two-thirds of the queries can be answered within 400ms, which is 16 times faster when compared to querying through the 3G link. From the search engine perspective, two-thirds of the query load can be eliminated resulting in significant cost savings and easier query load balancing during peak times. Finally, Section 2.7 discusses how future pocket cloudlets should be organized, from both an architectural and an operating system perspective and Section 2.8 discusses related work.

## 2.2 Potential Cached Services

NVM, in contrast to cellular bandwidth and battery power, is expected to become increasingly abundant in mobile devices. As described in Section 1.5.2, mobile phones are expected to soon reach 256 GB of NVM. It thus becomes very appealing to host cloud services directly on mobile devices, leveraging their fast increasing storage resources.

Dedicating only 10% of a 256 GB NVM to caching services results in 25.6 GB of available storage capacity. This storage could be used to augment various cloud services with a pocket cloudlets architecture, reducing the need for costly, slow and power-hungry long-range cellular communications. Table 2.1 shows the number of data items (*e.g.,* search result pages, web sites, etc.) that can be stored in 25.6 GB

Table 2.1: **Number of data items that can be stored in 25.6GB (10% of the projected NVM size available on low-end smartphones) for different pocket cloudlets.**

| **Pocket Cloudlet** | **Single Item** | **Number of Items** |
|---|---|---|
| Web Search | 100 KB (search result page) | ≈ 270,000 |
| Mobile Ads | 5 KB (ad banner) | ≈ 5,500,000 |
| Yellow Business | 5 KB (map tile with business info) | ≈ 5,500,000 |
| Web Content | 1.5 MB (`www.cnn.com`) | ≈ 17,500 |
| Mapping | 5 KB (128×128 pixels map tile) | ≈ 5,500,000 |

of space for various pocket cloudlets. A low-end smartphone is projected to be able to store more than 5 million map tiles or 17000 web sites. To put these numbers in perspective, our search log analysis indicated that more than 90% of mobile users visit fewer than 1000 URLs over a period of several months, which is 17 times fewer than the number of web sites that we can actually store on the phone. For the mapping service, assuming that each map tile covers $300 \times 300 m^2$ of actual earth surface, 5.5 million map tiles can cover the area of a whole state in the United States. As a result, the projected memory resources for smartphones could easily sustain the web browsing and mapping needs of a typical mobile user.

## 2.3 Pocket Cloudlet Architecture

Enabling mobile devices to efficiently host cloud services poses various challenges. First, the amount of data to be stored locally on the device needs to be determined for each cloud service. Even though the analysis in Section 1.5.2 shows that memory resources on mobile devices will be abundant in the next decade, the amount of data available on the Internet and across the different cloud services will normally far exceed the available memory resources on a typical smartphone. Second, a mechanism to manage the locally stored cloud data is required, as this data might change over time (*e.g.,* web content changes over time). Third, a storage architecture for efficiently

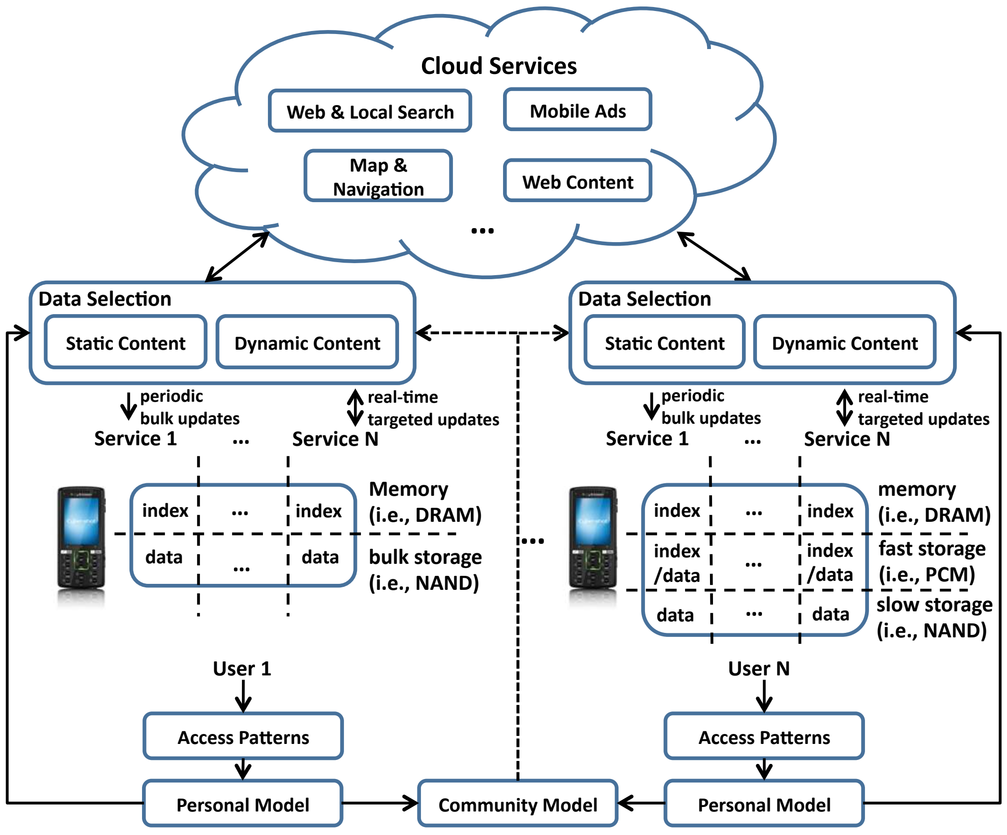


Figure 2.1: **The envisioned architectural support required to transform mobile devices into pocket cloudlets.**

storing and accessing this large amount of data is needed. Mobile users need to be able to quickly search and access data across services while still having enough space to store their personal data. Figure 2.1 shows the infrastructure required to transform mobile devices into pocket cloudlets.

### 2.3.1 Collaborative Data Selection

At a higher level, the data stored locally on the mobile device are selected based on both personal and community access models. The access patterns of the individual user to a specific service (*e.g.,* web search) are recorded and used to construct a personal model (*e.g.,* favorite search topics). At the same time, users collaborate by contributing their locally constructed personal models. The personal models across users are combined together into a community model that identifies the most popular parts of the cloud service data across all users. Both personal and community models are then used to identify the most frequently accessed parts of the cloud service data that is or might be of interest to the individual user.

### 2.3.2 Adaptation: Data Management

The Pocket Cloudlets architecture is an adaptive caching scheme. The locally cached cloud service data needs to get updated dynamically. Updates occur when the device has access to power resources and high bandwidth links (*e.g.,* charging and connected to Wi-Fi or tethered to a desktop computer).

Periodic updates (*e.g.,* nightly, weekly or monthly) based on the charging state of the device are appealing, but can only be effective for relatively static data. That is, data that is not very frequently updated. For instance, the search index or the map tiles used for search and mapping services are examples of static data that could be updated periodically, and only when the mobile device is charging, without hurting the quality of the cloud service.

However, not all cached data tend to be that static. For instance, web content, such as news and stock prices, is dynamic in nature and tends to be accessed by individual users several times within a day. For this type of pocket cloudlet, real-time updates over the radio link are required to guarantee freshness of the cached data.

Performing bulk updates over power-hungry and bandwidth-limited radio links is inefficient, if not impossible. Luckily, it turns out that the amount of dynamic data that is repeatedly accessed by mobile users tends to be small. For example, our analysis of 200 million mobile search queries submitted by hundreds of thousands of users (Section 2.4) showed that 70% of web visits tend to be revisits to less than a few tens of web pages for more than 50% of the users. As a result, instead of enforcing inefficient bulk updates over the radio link, only the small set of most frequently visited data (identified by the access patterns of the individual user) is updated in real-time.

### 2.3.3 Architectural Implications

At a lower level, each cloud service owns its own storage space on the mobile device and uses it to store the necessary data. For instance, the storage can mirror web pages in the case of the web content service, and map tile snapshots and local business information in the case of mapping and navigation services. Since the amount of data required by these services is expected to be large and should always be available on the device even after a power down, bulk non-volatile storage, such as NAND flash, is a suitable memory technology. Besides storing the actual data on the mobile device's NVM, each cloud service also maintains an index of its data in fast volatile memory (DRAM). The index enables instant retrieval of the required data from bulk storage. Given the characteristics of current memory technologies, the main memory of the phone (*i.e.,* DRAM) is able to provide the necessary performance and density for storing the different indexes.

As new memory technologies, such as PCM, mature, they could be used to store the index as well. The described two-tier memory structure would then evolve into a three-tier structure, as shown in Figure 2.1. PCM could become the intermediate tier by filling the performance and storage density gap between DRAM and NAND flash. In practice, PCM could be seen as fast bulk storage when compared to NAND flash, making it an ideal technology for storing the data indexes. While slower than DRAM, PCM has the advantage of being non-volatile and significantly faster than NAND flash. Being able to store data indexes in PCM eliminates the need to commit to and load the index from NAND flash after each power cycle of the mobile device. Given that the data required by the cloud services might be tens or even hundreds of gigabytes, the size of the data indexes can reach gigabytes, making its transfer between flash and main memory extremely time-consuming. By introducing a PCM-based layer, all data indexes could become instantly available on the device at boot time, offering a much faster user experience. At the same time, the DRAM tier could be used to cache the most frequently accessed parts of the data indexes in order to provide the fastest user experience when possible.

Figure 2.1 only shows the high-level architectural requirements to enable mobile cached cloud services. In practice, however, several lower-level challenges and design tradeoffs can arise. In the next several sections, we present these challenges and tradeoffs for an example cloud service that focuses on mobile search and advertisement.

## 2.4 Cacheability of Mobile Search

Before designing and implementing a pocket cloudlet for a cloud service, it is important to study its cacheability. For instance, the real impact of a search pocket cloudlet depends on the fraction of the query volume that can be successfully served locally on the device. To answer this question, we analyzed 200 million queries, submitted to `m.bing.com` over a period of several consecutive months in 2009. The query volume consisted of web search queries submitted from mobile devices, such as phones and personal media players. Every entry in the search logs we analyze contains the raw query string that was submitted by the mobile user as well as the search result that was selected as a result of the submitted query. No personal information, such as location, is included in the logs.

### 2.4.1 The Mobile Community Effect

First, we examine the community of mobile users as a whole to discover caching opportunities across users. From the web search logs, we extract the most popular queries submitted and the most popular search results clicked by the mobile users.

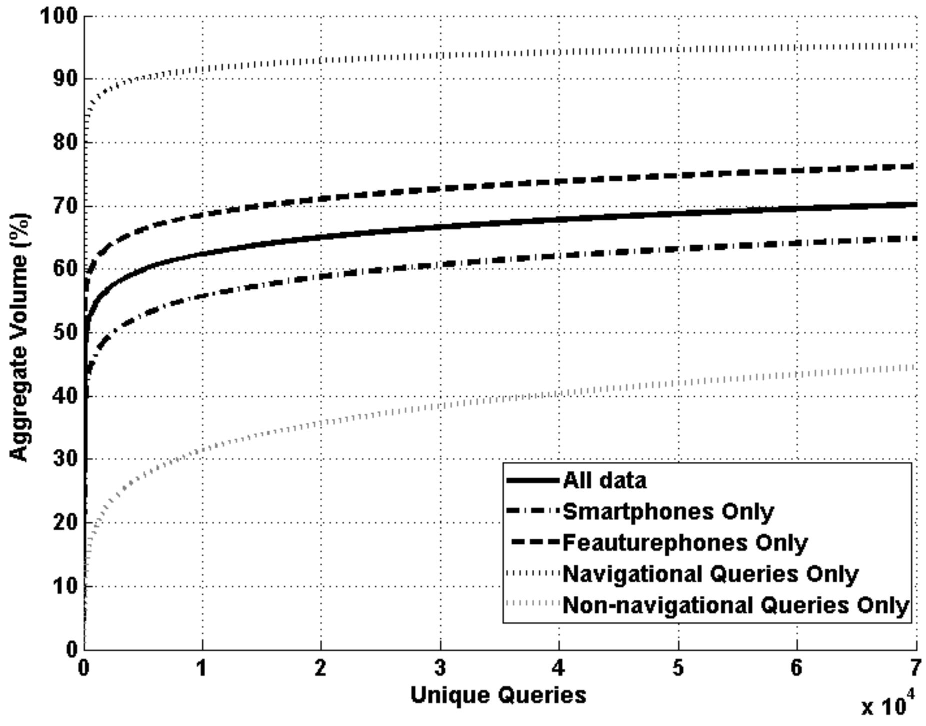
Figures 2.2(a) and (b) show the cumulative query and search result volume as a function of the number of most popular queries and search results, respectively. When looking across all data, it turns out that the 6000 most popular queries and the 4000 most popular search results are responsible for approximately 60% of the query and search result volumes, respectively. In other words, there is a small set of queries and search results that is popular across all mobile users. This suggests that if we store these 6000 queries and 4000 search results locally on the phone, we could theoretically serve 60% of the overall queries submitted by mobile users without having to use the radio link.

Note that these 4000 search results might not always point to static web pages. However, while some web content might be highly dynamic, the search results and queries that point to it can be relatively static. For instance, the CNN web page (`www.cnn.com`) is updated every minute and sometimes even more frequently. However, the way mobile users reach this dynamic web page is relatively static (*e.g.*, search for "cnn" or "news" and then click on the static search result that points to the CNN web page).

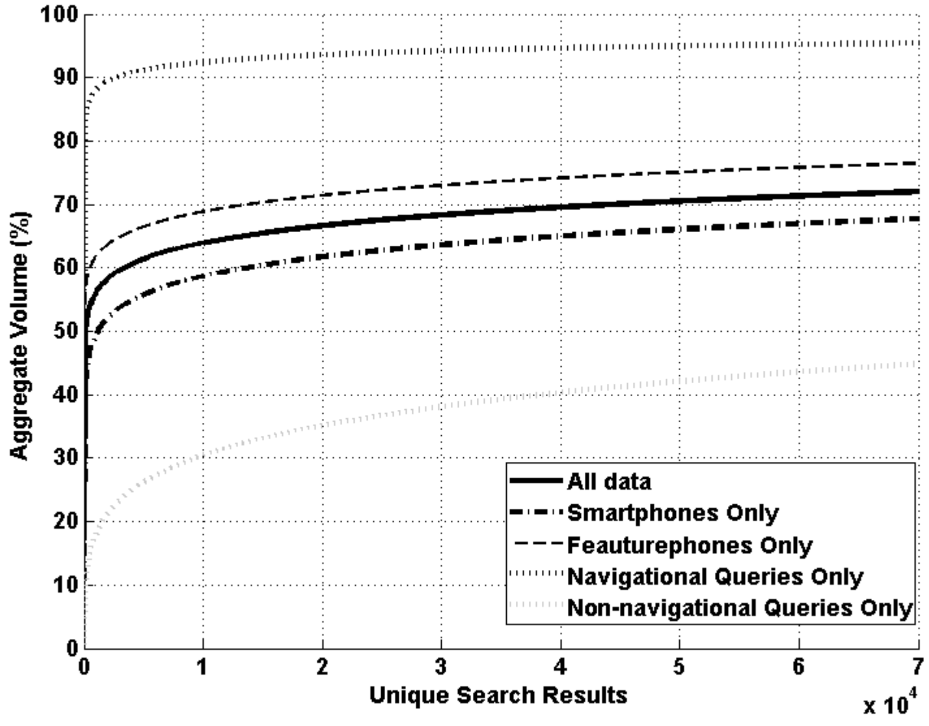
Similar popularity trends exist for desktop queries. However, mobile queries are significantly more concentrated than desktop queries. For instance, the first 6000 queries represent 60% of the query volume in the mobile domain, but less than 20% of the query volume in the desktop domain [72].

We further divide the queries in two different categories, navigational[1] (*e.g.,* "youtube" or "facebook") and non-navigational (*e.g.,* "michael jackson"), and we study the same trends for each category. As Figure 2.2 shows, both query types follow the same trends but navigational queries are significantly more concentrated compared to non-navigational queries. For instance, the first 5000 navigational queries are responsible for 90% of the navigational query volume while the same number of

[1]A query is classified as navigational when the actual query string is a substring of the clicked URL (*e.g.,* "youtube" and `www.youtube.com`)



(a)



(b)

Figure 2.2: **CDF plots of the (a) query volume and (b) clicked search result volume.**

non-navigational queries accounts for less than 30% of the non-navigational query volume.

Another interesting observation comes from comparing the results between Figures 2.2(a) and (b). To achieve a cumulative volume of 60%, 50% more queries are required compared to the number of search results (6000 queries vs. 4000 search results). The search logs show that users search for the same web page in many different ways. For instance, mobile users often either misspell their queries because of the small keyboards they have to interact with (*e.g.,* "yotube" instead of "youtube") or purposely change the query term to reduce typed characters (*e.g.,* "boa" instead of "bank of america"). However, even though a misspelled or altered query is submitted, the search engine successfully provides the correct search result and thus a successful click-through is recorded. As a result, a popular webpage is, in general, reached through multiple search queries.

Figures 2.2(a) and (b) also show the same information when considering the queries and search results that were submitted by feature phone (low-end mobile devices with limited browsers and Internet capabilities) and smartphone users in isolation. Even though the exact same observations hold in both cases, queries and search results that are accessed over feature phones are, in general, more concentrated when compared to smartphones. This is an artifact of the limited user interfaces found on feature phones that make web access a challenging task.

### 2.4.2 The Individual Mobile User Effect

Next, we examine personal query traces to discover caching opportunities within the search habits of individual users. In particular, we study how often individual users repeat queries. We call a query a repeated query only if the user submits the same query and clicks on the exact same search result. Figure 2.3 shows the percentage of individual mobile users as a function of the probability of submitting a new query

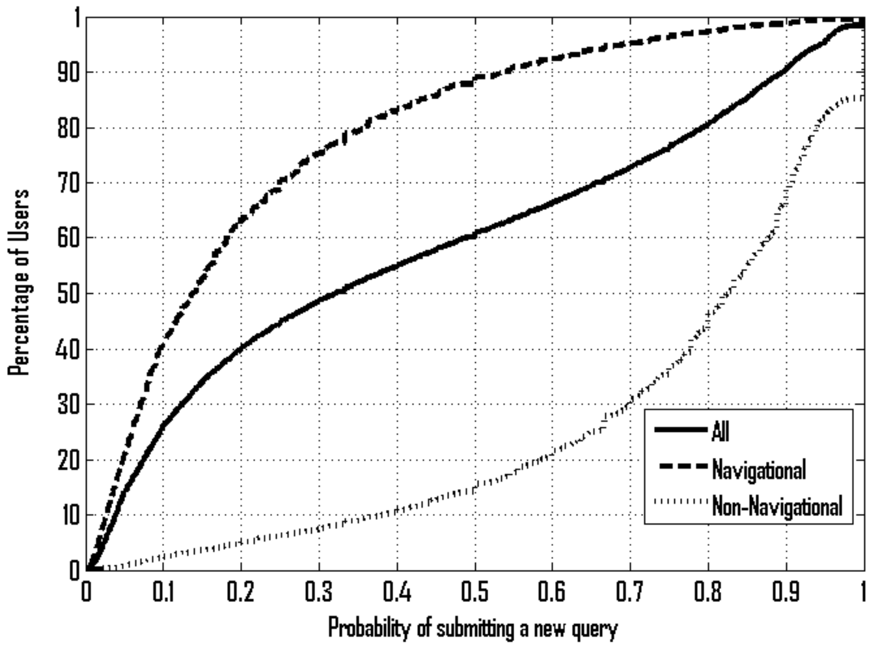


Figure 2.3: **CDF plot of the repeatability of mobile search queries across individual users over a 1-month period.**

within a month. Approximately 50% of mobile users will submit a new query at most 30% of the time. Thus, at least 70% of the queries submitted by half of the mobile users are repeated queries. Figure 2.3 also shows this trend for both navigational and non-navigational queries.

Consequently, knowing what the user searched for in the past provides a very good indication of what the user will search for in the future. Again, this trend of repeating queries is not unique to mobile queries. Desktop users also tend to repeat queries, but not as frequently as mobile users. Recent studies have shown that desktop users will repeat queries on average 40% of the time [142] as opposed to 56.5% for mobile users.

## 2.5 PocketSearch Architecture

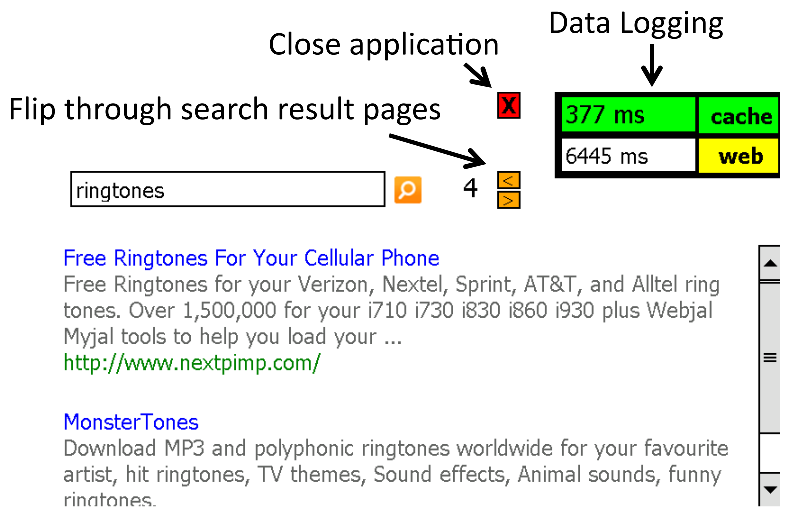
Given the mobile search trends that the analysis of the web search logs highlighted, we designed and implemented PocketSearch, a mobile search pocket cloudlet that lives on the phone and is able to answer queries locally without having to use long-range communications (3G link)[2]. The goal of the PocketSearch architecture is to capture the cacheability of mobile search as demonstrated in Section 2.4, and to be computationally tractable so that it can efficiently run on a mobile device. PocketSearch accomplishes both of these goals making its underlying architecture a template for other pocket cloudlet services.

Note that PocketSearch is not aiming to replace the actual search engine. Instead, its goal is to augment the search engine and help avoid the limitations of long-range communications in a way that is transparent to the user. Above all, PocketSearch helps provide a much faster user experience. As shown in Figure 2.4, while a query for the term "ringtones" took as long as $6445ms$ over 3G, it took only $377ms$ for PocketSearch.

PocketSearch can also make the user experience richer. Most of the high-end smartphones today can automatically provide query suggestions to the user almost instantly and as the user is typing his query. PocketSearch's ability to retrieve search results fast can make this experience richer by enabling the display of actual search results along with auto-suggest query terms in the auto-suggest box in real-time. This is shown in Figure 2.5 for our local search/ads prototype. If users are not interested in any of these search results, they can access the latest set of search results through the 3G radio by selecting the query term provided by auto-suggest or by entering the full query term on their own.

As shown in Figure 2.6, PocketSearch consists of two discrete but strongly interrelated components; the community and the personalization components. The community part of the cache is responsible for storing the small set of queries and search results that are popular across all mobile users. This information is automat-

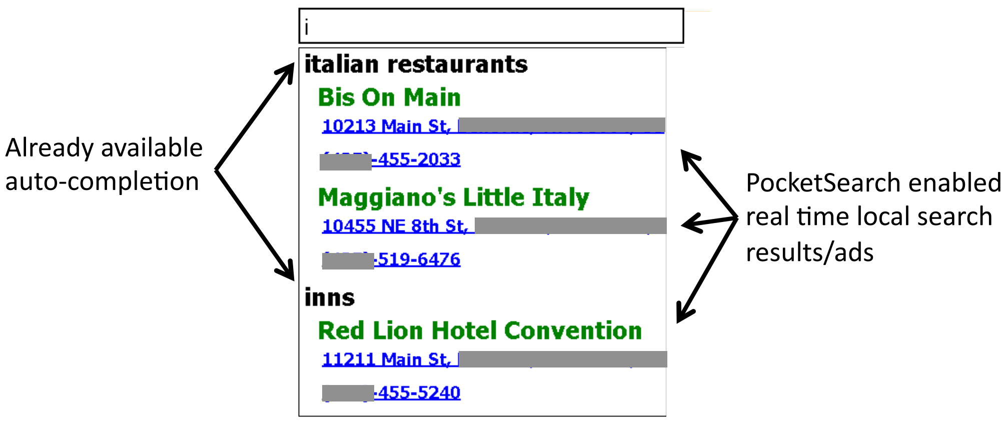
[2]PocketSearch only stores search results and not the actual web content that these results point to. Another cloudlet responsible for web content caching/pre-fetching (*i.e.*, PocketWeb) running along with PocketSearch could be used to serve the actual web content.



Free Ringtones For Your Cellular Phone
Free Ringtones for your Verizon, Nextel, Sprint, AT&T, and Alltel ring tones. Over 1,500,000 for your i710 i730 i830 i860 i930 plus Webjal Myjal tools to help you load your ...
http://www.nextpimp.com/

MonsterTones
Download MP3 and polyphonic ringtones worldwide for your favourite artist, hit ringtones, TV themes, Sound effects, Animal sounds, funny ringtones.

Figure 2.4: **The GUI of the PocketSearch prototype for web search. In this example, cached search results for the query "ringtones" were displayed in 377*ms*. The same query, when submitted over 3G, took 6445*ms* to complete.**


Already available auto-completion
PocketSearch enabled real time local search results/ads
i

italian restaurants
Bis On Main
10213 Main St, [redacted]
[redacted]-455-2033
Maggiano's Little Italy
10455 NE 8th St, [redacted]
[redacted]-519-6476
inns
Red Lion Hotel Convention
11211 Main St, [redacted]
[redacted]-455-5240

Figure 2.5: **The GUI of the PocketSearch prototype for local search/ads. Local search results/ads are instantly displayed in the auto-suggest box as the user types the query.**

ically extracted from the search logs. The search logs come from two sources: the actual search logs that search engines maintain and the personal user access patterns (logs) that users collaboratively contribute. Note that the latter is necessary since queries that are served locally by the pocket cloudlet do not appear in the former. The community part of the cache is updated overnight every time the mobile device is

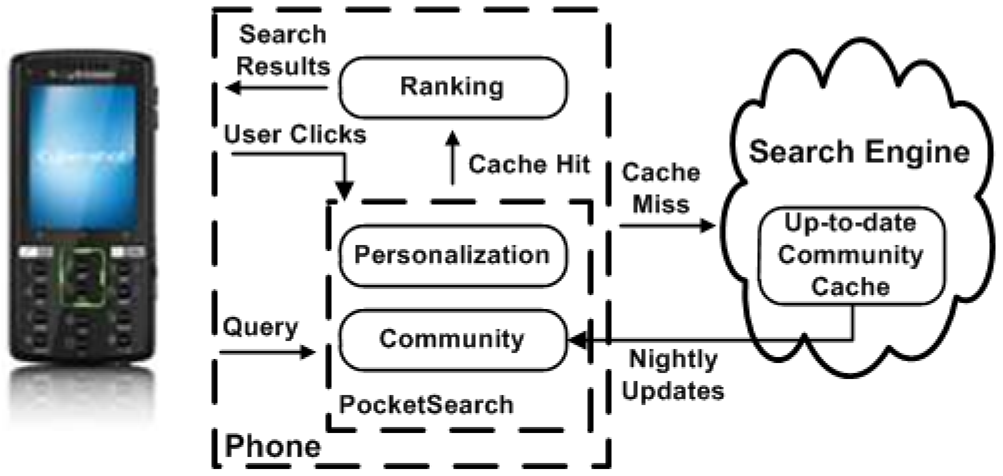


Figure 2.6: **Overview of the PocketSearch cloudlet.**

recharging, making sure that the latest popular information is available on the mobile device. The community part serves as a warm start for the cache and enables PocketSearch to instantly provide search results without requiring any previous knowledge of the user.

The personalization part of the cache monitors the queries entered as well as the search results clicked by the user and adapts accordingly, performing two discrete tasks. First, it expands the cache to include all those queries and search results accessed by the user that did not initially exist in the community part of the cache. In that way, the cache can take advantage of the repeatability of the queries submitted by the mobile users to serve as many queries as possible locally on the mobile device. Second, it collects information about user clicks, such as when and how many times the user clicks on a search result after a query is submitted, to customize ranking of search results to user's click history.

When a query is submitted, PocketSearch will first perform a lookup in the cache to find out if there are locally-available search results for the given query. In the case of a cache hit, the search results are fetched from the local storage, ranked based on the past user access patterns recorded by the personalization part of the cache, and immediately displayed to the user. In the case of a cache miss, the query is submitted to the search engine over the 3G radio link.

Realizing the architecture shown in Figure 2.6 on an actual mobile device poses several challenges:

**Content Generation**: A methodology is required to decide which and how many queries and search results should be included in the cache.

**Storage Architecture**: An efficient way to store and quickly retrieve the search results on the mobile device is needed. Memory overhead should be minimized to prevent performance degradation on the device and provide ample storage space for user's personal files. At the same time, PocketSearch should be able to quickly locate and retrieve the search results to minimize user response time.

**Personalized Ranking**: The user's search patterns provide important information about the individual user's interests. PocketSearch should record and leverage this information over time to adapt its cache and personalize the search experience.

**Cache Management**: A scalable mechanism for adaptively updating the cache contents is required. Having available the most up-to-date set of popular queries and search results on the phone is necessary for maximizing the number of queries that can be served by PocketSearch.

### 2.5.1 Cache Content Generation

The search results stored in the cache are extracted directly from the combined mobile search logs (search engine logs, contributed personal user logs). The goal of this process is to identify the most popular queries and search results that are of interest to the mobile community.

A set of triplets in the form <query, search result, volume> are extracted from the search logs and sorted based on *volume* (Table 2.2). The term *query* corresponds to the query string submitted to the search engine, the term *search result* corresponds

Table 2.2: **A list of query-search result pairs sorted by their volume is generated by processing the mobile web search logs over a time window (*e.g.,* a month). The volume numbers used in this table are hypothetical.**

| **Query** | **Search Result** | **Volume** |
|---|---|---|
| michael jackson | `www.imdb.com/name/nm0001391/bio` | $10^6$ |
| movies | `www.fandango.com` | $95 * 10^4$ |
| michael jackson | `www.azlyrics.com/j/jackson.html` | $90 * 10^4$ |
| ringtones | `www.myxer.com` | $50 * 10^4$ |
| pof | `www.plentyoffish.com` | $20 * 10^4$ |
| ... | ... | ... |
| **Total Volume** | | $50 * 10^5$ |

to the search result that was selected after entering the query, and the term *volume* represents the number of times in the search logs that the specific *search result* was selected after entering the query string *query*. For instance, the first row in Table 2.2 can be interpreted as follows: *In the last month, there were 1 million searches where the search result www.imdb.com/name/nm0001391/bio was selected after the query "michael jackson" was submitted.*

The number of triplets in Table 2.2 can be in the order of tens or hundreds of millions. Storing all of them would require significant memory resources that a phone might not be able to provide or a user might not be willing to sacrifice. Therefore, we need to implement a policy for selecting which items to store. To maximize the query volume that can be served by the cache, we should always store the most popular pairs of queries and search results indicated by the top entries of Table 2.2.

Deciding how many of the most popular *query-search result* pairs to store is a more complicated process. We select the number of *query-search result* pairs to cache based on either a memory or cache saturation threshold, as described next.

**Memory (NAND flash or DRAM) Threshold**: Starting from the top entry in Table 2.2, we run down through its entries and continuously add *query-search result* pairs until a specific flash or DRAM memory size threshold $M_{th}$ is reached. This

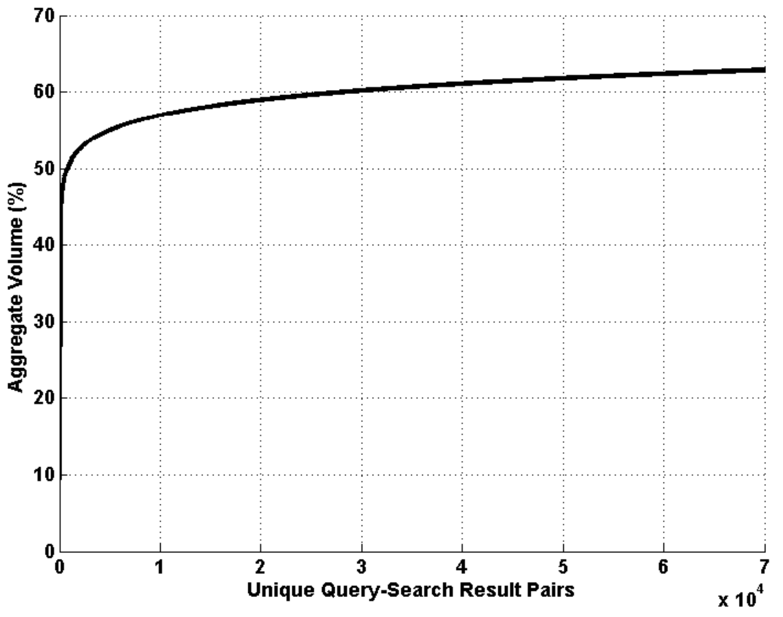


Figure 2.7: **Cumulative query-search result volume as a function of the most popular query-search result pairs.**

threshold can be set by either the phone itself based on its available memory resources or by the user, depending on how much storage space and memory the user is willing to sacrifice for PocketSearch.

**Cache Saturation Threshold**: Starting from the top entry in Table 2.2, we run down through its entries and continuously add *query-search result* pairs until we reach a *query-search result* pair with a normalized volume lower than a predetermined threshold $V_{th}$. The normalized volume of a *query-search result* pair is generated by dividing this pair's volume by the total volume of all *query-search result* pairs in the search logs. For instance, the *normalized volume* of the first query-search result pair in Table 2.2 is equal to: $10^6/(5 * 10^6) = 0.2$.

The value of the cache saturation threshold is illustrated in Figure 2.7. It is apparent that the value of adding *query-search result* pairs quickly diminishes. In particular, slightly increasing the aggregate volume from 58% to 62% requires doubling the amount of *query-search result* pairs from 20000 to 40000.

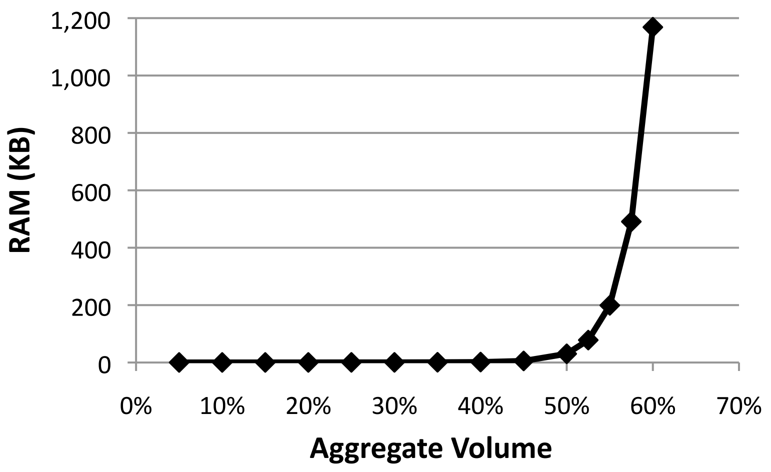


Figure 2.8: **PocketSearch's DRAM overhead for different *query-search result* aggregate volumes.**

In practice, the mobile web search log analysis we performed showed that the cache saturation threshold will be quickly reached before PocketSearch stretches the memory or storage resources available on the phone. This can be seen in Figures 2.8 and 2.9 that show the size of DRAM and flash, respectively, required by PocketSearch as a function of the aggregate *query-search result* volume represented by all the pairs stored in the cache. It is clear that the saturation point of the cache is quickly reached when the most popular *query-search result* pairs that correspond to approximately 55% of the cumulative *query-search result* volume have been cached. At this point, the cache requires approximately 1MB of flash and 200KB of DRAM, which accounts for less than 1% of the available memory and storage resources on a typical smartphone.

Independently of which threshold is used (memory or cache saturation), this methodology identifies the $n$ top entries in Table 2.2. Each of these $n$ top *query-search result* pairs is then associated with a ranking score that is produced by normalizing its volume across all search results that correspond to the query. For instance, in the case of query "michael jackson" in Table 2.2, the ranking score for the *imdb* search result is equal to $10^6/(1.9*10^6) = 0.53$ and the score for the *azlyrics* is $9*10^5/(1.9*10^6) = 0.47$.

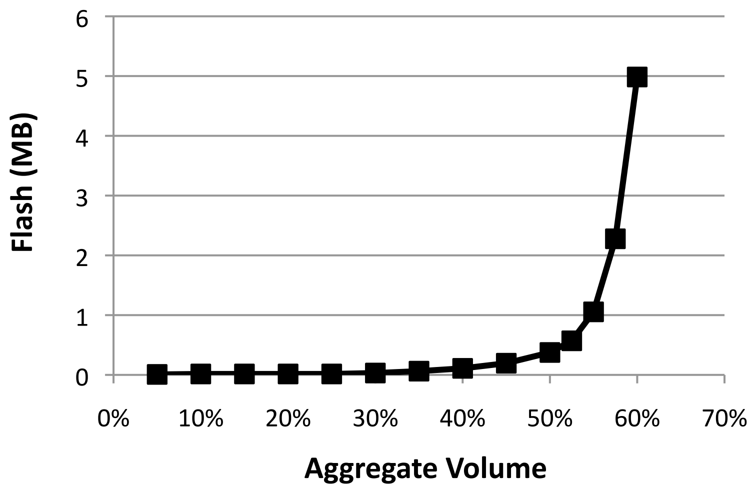


Figure 2.9: **PocketSearch's flash overhead for different *query-search result* aggregate volumes.**

The generated <query, search result, score> triplets can now be used to build the cache on the phone.

Extracting PocketSearch's cache contents directly from the mobile search logs provides several advantages. First, even though there might be tens or even hundreds of search results available for a given query, we only cache these search results that are popular across all mobile users, limiting the amount of memory resources required. Second, each query and search result pair extracted from the search logs is associated to a ranking score, enabling the phone to rank search results locally. Third, by processing the mobile search logs, we automatically discover the most common misspellings and shortcuts of popular queries, enabling PocketSearch to cache search results for these cases. As a result, queries such as "pof", and "boa" can now be served locally on the phone by instantly displaying search results such as `www.plentyoffish.com` and `www.bankofamerica.com`, respectively.

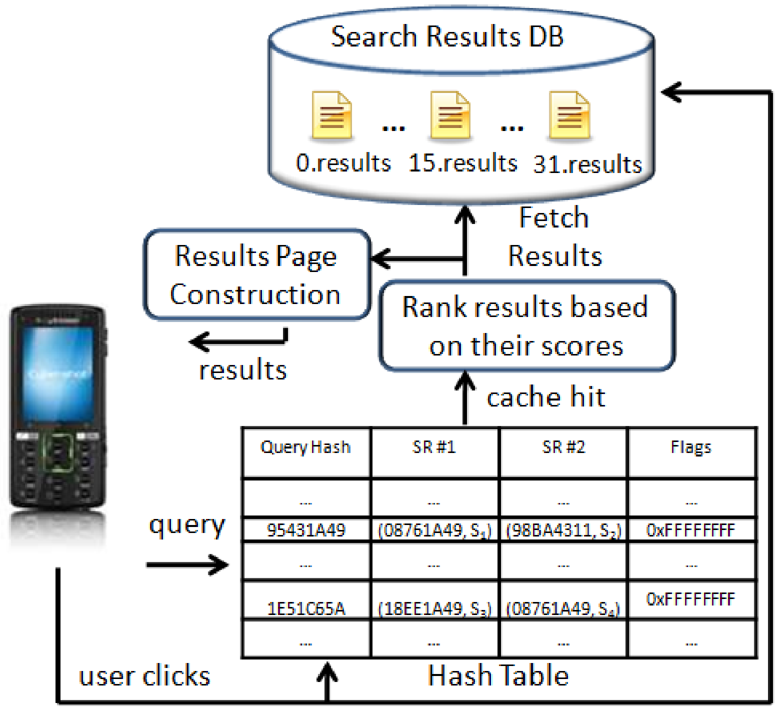
### 2.5.2 Storage Architecture

The extracted set of <query, search result, score> triplets must be efficiently stored on the phone. Storage efficiency is defined in two ways. First, the memory resources required to store the search results should be as low as possible to permit many pocket cloudlet services to concurrently run. Second, the time it takes to retrieve, rank and display search results after the user enters a query should be as low as possible.

Figure 2.10 provides an overview of PocketSearch's storage architecture. It consists of two components, a hash table and a custom database of search results. The hash table lives in main memory and its role is to link queries to search results. Given a query, the hash table can quickly identify if we have a cache hit or a cache miss. In the case of a cache hit, the hash table provides pointers to the database where the search results for the submitted query are located. Along with each search result pointer, the hash table provides its ranking score, enabling PocketSearch to properly rank search results on the phone.

The custom database of search results resides in flash and its role is to store all the available search results so that they occupy the least possible space and they can be quickly retrieved. The data stored in the database for each search result includes all the necessary information for generating the same search user experience with the search engine: the actual web address, a short description of the website and the human readable form of the web address.

Over time and as the user submits queries and clicks on search results, PocketSearch updates both the hash table and the database of search results. Every time the user clicks on a search result, its ranking score is properly updated in the hash table. In addition, if a new query or a new search result is selected that does not exist in the cache, both the hash table and the database are properly updated so that this query and search result can be retrieved from the cache in the future.

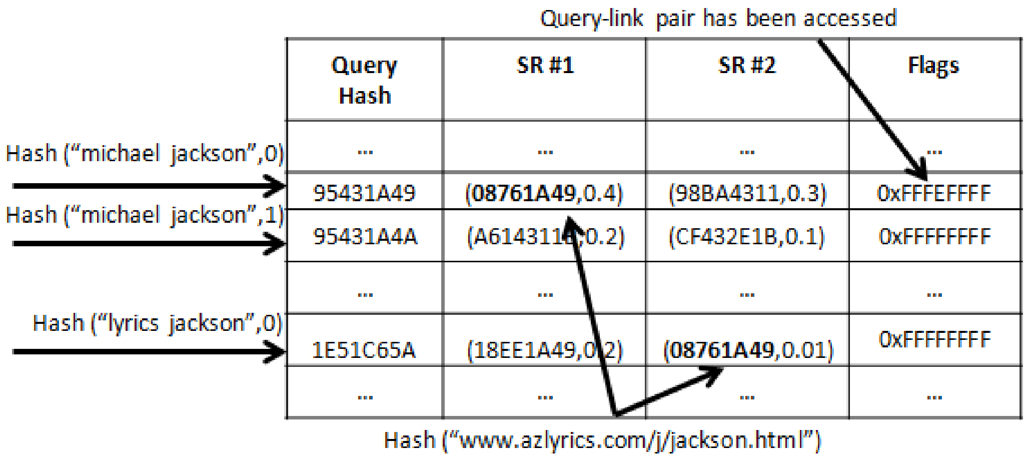


| Query Hash | SR #1 | SR #2 | Flags |
|---|---|---|---|
| ... | ... | ... | ... |
| 95431A49 | (08761A49, $S_1$) | (98BA4311, $S_2$) | 0xFFFFFFFF |
| ... | ... | ... | ... |
| 1E51C65A | (18EE1A49, $S_3$) | (08761A49, $S_4$) | 0xFFFFFFFF |
| ... | ... | ... | ... |

Figure 2.10: **Overview of PocketSearch's storage architecture.**

### Query Hash Table

Figure 2.11 shows the structure of the hash table used to link queries to search results. Every entry in the hash table corresponds to one and only one query and has four fields. The first field contains the hash value of the query string that this entry corresponds to. The next two fields are of identical type and represent two search results associated with the query (SR #1 and SR #2 in Figure 2.11.). As explained later in detail, only two search results are stored per entry to minimize hash table's memory footprint. Each search result in the hash table is represented by a pair of numbers. The first number corresponds to the hash value of the web address of the search result. This value is used to uniquely identify a search result and, as described in the next section, is used as a pointer to retrieve the information associated with the search result (short description, web address, etc.) from the database. The second number corresponds to the ranking score of the search result. The last field of each entry in the hash table


Query-link pair has been accessed
Hash ("michael jackson",0)
Hash ("michael jackson",1)
Hash ("lyrics jackson",0)
Hash ("www.azlyrics.com/j/jackson.html")

| Query Hash | SR #1 | SR #2 | Flags |
|---|---|---|---|
| ... | ... | ... | ... |
| 95431A49 | (**08761A49**,0.4) | (98BA4311,0.3) | 0xFFFEFFFF |
| 95431A4A | (A614311[illegible],0.2) | (CF432E1B,0.1) | 0xFFFFFFFF |
| ... | ... | ... | ... |
| 1E51C65A | (18EE1A49,0.[illegible]2) | (**08761A49**,0.01) | 0xFFFFFFFF |
| ... | ... | ... | ... |

Figure 2.11: **The hash table data structure used to link queries to search results.**

is a 64-bit number that is used to log information about the two search results in this entry. Currently, we use only one bit for each search result to indicate if the user has ever accessed the specific *query-search result* pair. The rest of the flag bits are reserved for future purposes.

In general, given a set of <query, search result, score> triplets, the hash table is generated as follows. For every unique query in the set of triplets, we identify all the search results associated with this query. An entry is created in the hash table for the query and search results are added in descending order of score. If more than two search results are associated with the same query, additional entries are created in the hash table by properly setting the second argument of the hash function (*e.g.,* "michael jackson" query in Figure 2.11).

This approach of linking queries to search results highlights two important design decisions that were influenced by the properties of the <query, search result, score> triplets extracted from the mobile web search logs. First, the number of search results linked to a query in a hash table entry affects the memory footprint of the hash table. This is illustrated in Figure 2.12 that shows the memory footprint of the hash table for

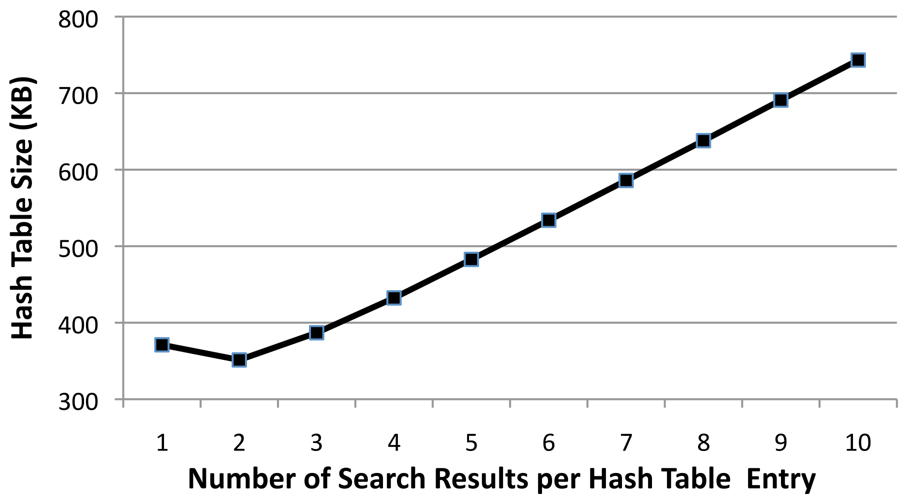
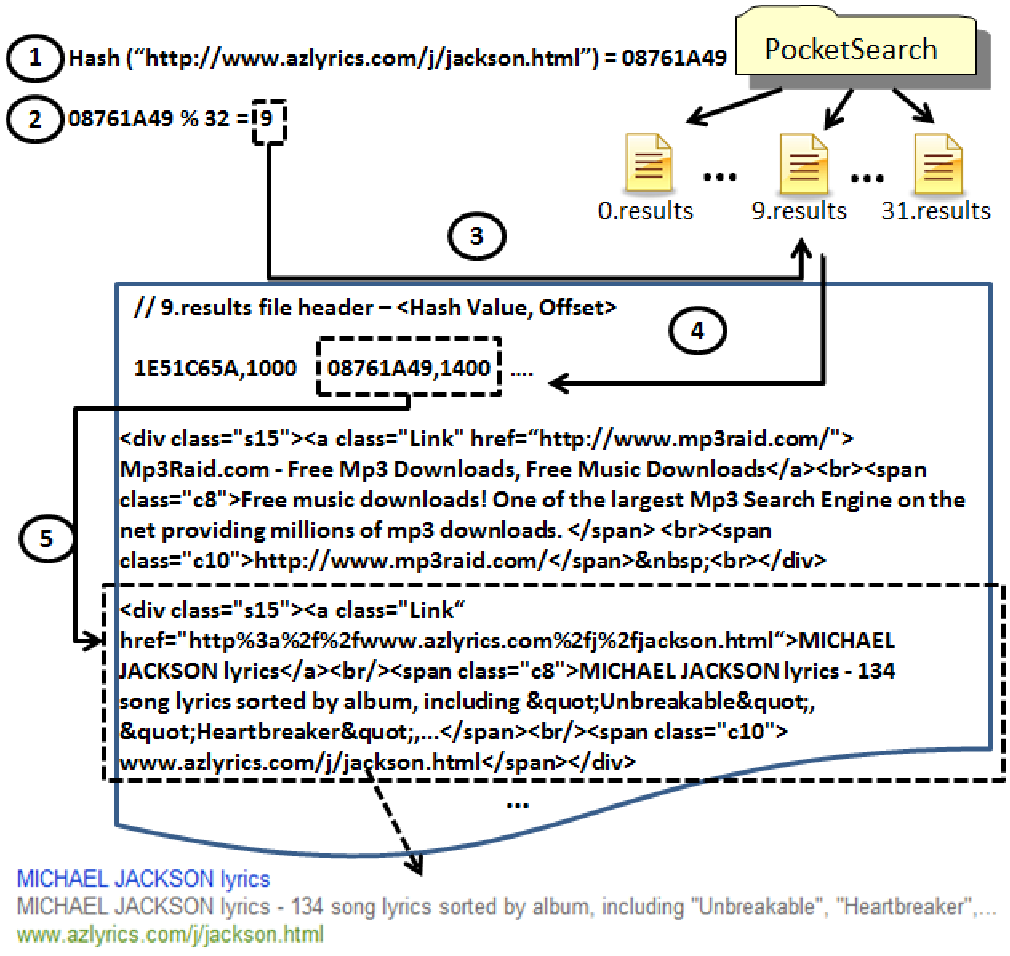


Figure 2.12: **The memory footprint of the hash table for different number of search results per hash table entry.**

different numbers of search results stored per hash table entry. The smallest memory footprint is achieved when two search results are stored per hash table entry.

Second, the way queries are linked to search results can affect the storage requirements of the database of search results. The simplest and fastest approach to retrieving and displaying search results to the user would be to store them in a single HTML file. Even though this approach would simplify the structure of the hash table, it would significantly increase the flash memory required to store them. The reason is that most of the search results are shared across a large number of queries. The analysis of the logs indicates that only 60% of the search results in PocketSearch are unique. If a single search result page were to be stored for every query, then 40% of the search results would have to be stored at least twice. To avoid wasting flash resources, we opted to store each search result once and then link individual queries to each search result independently. Besides saving space, this approach also enables PocketSearch to easily add/remove search results to/from the hash table and update the ranking score of search results over time. As Section 2.6 shows, the overhead introduced by this approach in terms of user response time is negligible.


1 Hash (“http://www.azlyrics.com/j/jackson.html”) = 08761A49
2 08761A49 % 32 = 9
PocketSearch
0.results
9.results
31.results
3
4
5
// 9.results file header – <Hash Value, Offset>
1E51C65A,1000 08761A49,1400 ....

```
<div class="s15"><a class="Link" href="http://www.mp3raid.com/">
Mp3Raid.com - Free Mp3 Downloads, Free Music Downloads</a><br><span
class="c8">Free music downloads! One of the largest Mp3 Search Engine on the
net providing millions of mp3 downloads. </span> <br><span
class="c10">http://www.mp3raid.com/</span>&nbsp;<br></div>
<div class="s15"><a class="Link"
href="http%3a%2f%2fwww.azlyrics.com%2fj%2fjackson.html">MICHAEL
JACKSON lyrics</a><br/><span class="c8">MICHAEL JACKSON lyrics - 134
song lyrics sorted by album, including &quot;Unbreakable&quot;,
&quot;Heartbreaker&quot;,...</span><br/><span class="c10">
www.azlyrics.com/j/jackson.html</span></div>
...
```

MICHAEL JACKSON lyrics
MICHAEL JACKSON lyrics - 134 song lyrics sorted by album, including "Unbreakable", "Heartbreaker",...
www.azlyrics.com/j/jackson.html

Figure 2.13: **The 32-file custom database and illustration of the process of retrieving a search result.**

### Search Results Organization

Search results are stored in flash using a custom database of plain text files to ensure portability of PocketSearch across different mobile platforms. For every search result, we store its title, which serves as the hyperlink to the landing page, a short description of the landing page and the human readable form of the hyperlink (Figure 2.13).

The amount of memory required to store the information associated to a search result into a file is, on average, 500 bytes. However, the actual memory space required might be significantly higher due to the internal structure of flash chips. Flash memories are organized in blocks of fixed size that are usually equal to 2KB, 4KB or

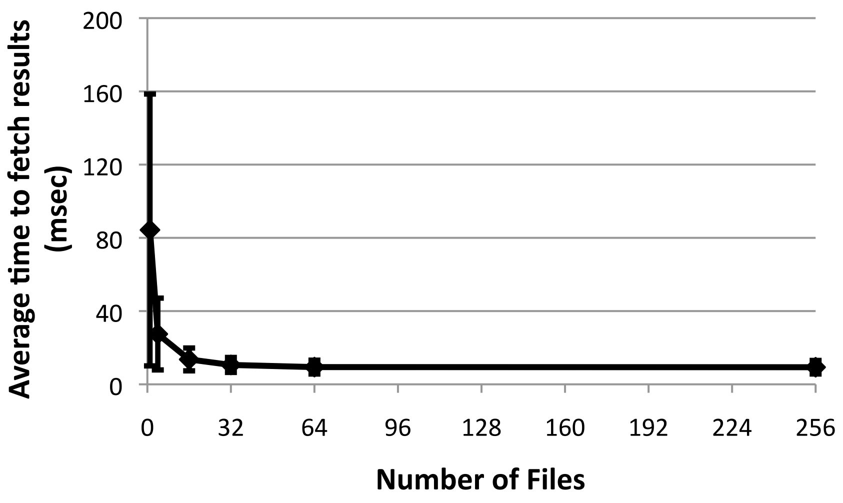


Figure 2.14: **Average time to retrieve two search results from the database as a function of the number of files used to store the search results. The vertical bars represent the deviation of the access time over 10 consecutive experiments.**

8KB depending on the size of the chip. If we store a 500-byte file containing a single search result in flash memory, then this file will occupy 4, 8 or 16 times more flash space than its actual size depending on the block size that is used.

In order to avoid flash fragmentation, multiple search results should be aggregated and stored into as few files as possible. However, storing a large number of search results into a single file could increase the time it takes PocketSearch to locate and retrieve a search result and, thus, it could hurt the response time of the cache. As a result, the way search results are aggregated into files and organized within a file is critical for minimizing both, flash fragmentation and cache response time. By evaluating different database organizations (Figure 2.14), we found that a number of 32 database files constitutes the best tradeoff between flash fragmentation and user response time.

Figure 2.13 shows how search results are organized within a database file when 32 database files are used. Each search result is assigned to one of the 32 files based on the hash value of its web address. In particular, the remainder of the division of the hash value with the number of files in the database (a number between 0 and 31) is used to identify the file where the search result should be stored.

The first line in each of the 32 database files contains pairs of the form $(hash\,value, offset)$. The *offset* represents the actual offset from the beginning of the file where the information for the search result represented by the *hash value* is located. By parsing the first line of a database file we can identify where each search result stored in this file is located. Whenever the user clicks on a search result that is not already cached, PocketSearch will add the search result at the end of the database file and augment the header of this file with the $(hash\,value, offset)$ pair for this search result.

### 2.5.3 Personalized Ranking

By monitoring user clicks over time, the personalization component of the cache is aware of when and how many times the user selects a search result after a given query is submitted. PocketSearch uses this information to incrementally update the ranking score of the cached search results to offer a personalized search experience.

Assume that for a query $Q$, there are two search results $R_1$ and $R_2$ available in the cache. Every time the user submits the query $Q$ and clicks on the search result $R_1$, PocketSearch updates the scores $S_1$ and $S_2$, for the two search results $R_1$ and $R_2$, respectively, as follows:

$$S_1 = S_1 + 1 \tag{2.1}$$

$$S_2 = S_2 * e^{-\lambda} \tag{2.2}$$

The ranking score of the selected search result is increased by 1 (Equation 2.1), the maximum possible score of a search result extracted from the mobile search logs. In that way, we always favor search results that the user has selected. Note that if this search result did not initially exist in the cache (selected after a cache miss), then a new entry in the hash table is created that links the submitted query to the selected search result and its score becomes equal to 1. At the same time, the ranking score for the unselected search result is exponentially decreased [3] (Equation 2.2). This enables PocketSearch to take into account the freshness of user clicks. For instance, if search result $R_1$ was clicked 100 times one month ago and search result $R_2$ was clicked 100 times during last week, then the ranking score for $R_2$ will be higher.

Using Equations (2.1) and (2.2), the ranking score of the search results, at any given time, reflects both the number and freshness of past user clicks. In practice, any personalization ranking algorithm [142, 143] could be used with the proposed cache.

### 2.5.4 Cache Management

Figure 2.15 provides an overview of the mechanism used to adaptively manage the community component of the cache. The phone transmits to the server its current version of the hash table. The server runs through the hash table and removes all the *query-search result* pairs that have not been accessed by the user in the past. This can be easily done by examining the *flags* column in the hash table (Figure 2.11). The *query-search result* pairs that have been accessed by the user in the past are only removed from the cache when their ranking score becomes lower than a predetermined threshold (*e.g.,* the user has not accessed the search result over the last three months).

At the same time, the server dynamically (*e.g.,* daily, when a new trend is detected *etc.*) extracts the most popular queries and search results from the combined mobile search logs (search engine logs, contributed personal user logs), as described in Section 2.5.1, and adds them to the hash table. During this process, conflicts might arise in the sense that a *query-search result* pair that already exists in the hash table (previously accessed by the user) might re-appear in the popular set of queries and search results extracted on the server. The conflict is caused when the ranking score stored in the

[3] The parameter $\lambda$ controls how fast the ranking score is decayed.

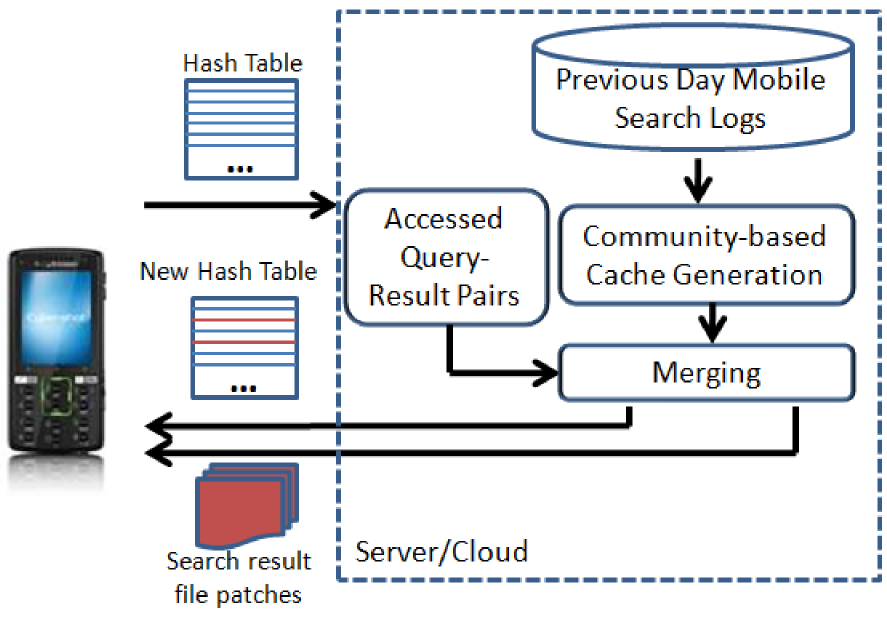

Hash Table
New Hash Table
Search result file patches
Previous Day Mobile Search Logs
Accessed Query-Result Pairs
Community-based Cache Generation
Merging
Server/Cloud

Figure 2.15: **Overview of PocketSearch's updating process.**

hash table is different from the new ranking score computed on the server based on the search log analysis. PocketSearch resolves these conflicts by always adopting the maximum ranking score.

After the hash table has been updated, the server creates the necessary patch files for the database files that live on the phone. The new hash table and the 32 patch files are transmitted to the phone and the new cache becomes available to the user. Note that the amount of data exchanged between the phone and the server will usually be less than 1.5 MB, given that PocketSearch requires, on average, approximately 200 KB for storing the hash table (Figure 2.8) and 1MB for storing the search results (Figure 2.9).

## 2.6 PocketSearch Evaluation

Pocket cloudlets, in general, and PocketSearch, in particular, by responding to service requests locally and reducing the need for long-range cellular communication result in reduced cost, lower energy consumption, lower response time and lower bandwidth usage. In order to evaluate the benefits of PocketSearch, we first quantify the amount of time and energy required to serve a search query through PocketSearch and compare its performance to that of the different radio links available on the phone. Second, we extract anonymized search query streams from the `m.bing.com` search logs and run them against PocketSearch to quantify what fraction of the query volume of an actual user can be served locally on the phone.

### 2.6.1 Cache Hit Performance

All measurements presented in this section were acquired using the prototype PocketSearch implementation on a Sony Ericsson Experia X1a cell phone running Windows Mobile 6.1 connected to AT&T's network. To measure how fast queries are served using PocketSearch and the different radios available on the phone (EDGE, 3G[4], 802.11g), we randomly selected 100 different queries for which cached search results were available. In every experiment, each of the 100 queries was submitted 100 times and the average user response time was calculated. The user response time is defined as the elapsed time from the moment the query is submitted to the moment the embedded browser object in the application has completed rendering the search results web page.

In the experiments where the PocketSearch cache was used to serve queries, a cache containing all the *query-search result* pairs that account for 55% of the cumulative *query-search result* volume over a period of several months was used. This cache included approximately 2500 search results occupying 1 MB of flash space, as described in Section 2.5.2.

[4]3G is the commonly-used commercial name. More specifically, the HSPA protocol was used.

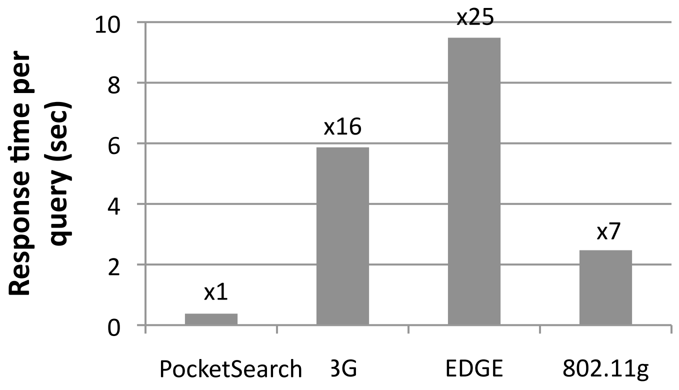


Figure 2.16: **Average search user response time per query.**

### Search User Response Time

Figure 2.16 shows the average user response time per query when the PocketSearch cache or one of the radios on the phone is used. On average, PocketSearch is able to serve a query 16 times faster than 3G, 25 times faster than EDGE and 7 times faster than 802.11g. Note that even though 802.11g can provide a low user response time that is slightly higher than 2 seconds, it has a major drawback. Due to its high power consumption, 802.11g is rarely turned on and connected to an access point on a continuous basis. As a result, in practice, 802.11g is not instantly available and requires extra steps that introduce delay and unnecessary user interaction.

Table 2.3 shows the breakdown of PocketSearch's user response time in the case of a cache hit. 96.7% of the time it takes PocketSearch to serve a query is spent at the browser while rendering the search results web page. The time it takes the cache to locate and retrieve search results is approximately 10ms and it accounts for only 3.3% of the overall user response time.

Furthermore, the time it takes PocketSearch to look up its hash table and determine if a query is a cache hit or a cache miss is only $10 \mu s$. Therefore, in the case of a cache miss, the overall user response time will only be increased by $10 \mu s$, a negligible

Table 2.3: **PocketSearch user response time breakdown.**

| Operation | Average Time (ms) | Percentage |
|---|---|---|
| **Hash Table Lookup** | 0.01 | $\approx 0\%$ |
| **Fetch Search Results** | 10 | 2.7% |
| **Browser Rendering** | 361 | 96.7% |
| **Miscellaneous** | 7 | 1.7% |
| **Total** | 378 | 100% |

Table 2.4: **Navigation user response time for PocketSearch and 3G for two example webpage load times.**

| | **Navigation user response time** | | |
|---|---|---|---|
| | **PocketSearch** | **3G** | **Speedup over 3G** |
| **Lightweight Page** | 15.378s | 21.048s | 28.7% |
| **Heavyweight Page** | 30.378s | 36.048s | 16.7% |

increase given that any radio on the phone requires several seconds to serve a search query.

### Navigation User Response Time

By reducing search user response time, PocketSearch manages to also improve the overall navigation user response time that includes both the time it takes the user to first search the web as well as the time to download the actual webpage. Table 2.4 shows the actual navigation time for two representative pages, a lightweight version and a heavyweight version that take 15 seconds and 30 seconds, respectively, to be downloaded and rendered over 3G. When PocketSearch serves the search results, the user can access the desired webpage up to approximately 29% faster than when querying through the 3G link.

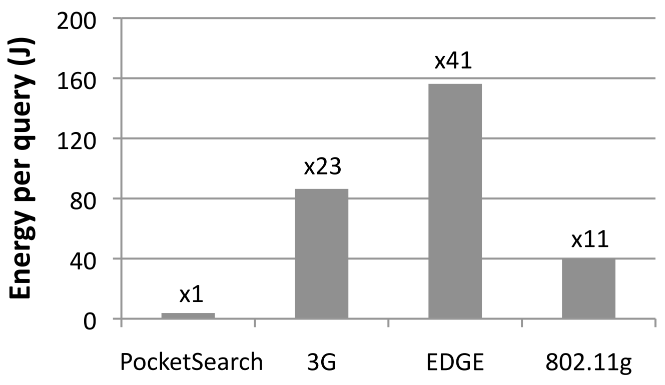


Figure 2.17: **Average energy per query.**

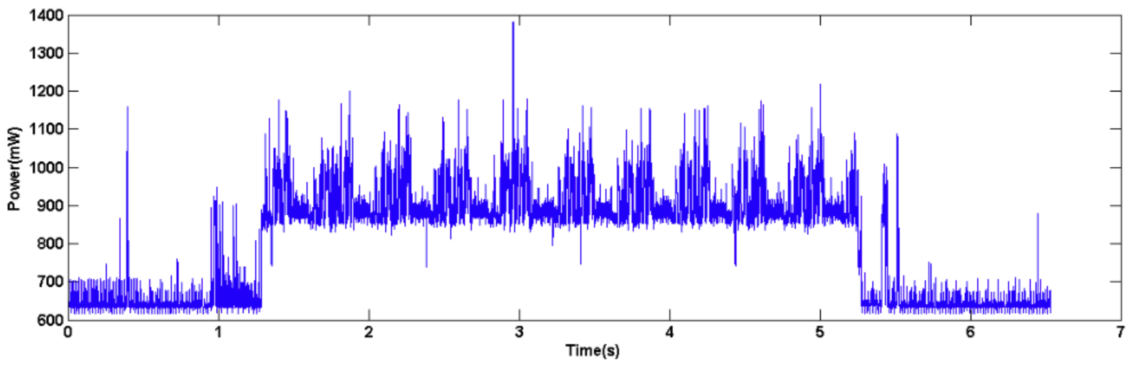
**Energy Consumption**

Figure 2.17 shows the average energy consumed by the phone per query when PocketSearch or one of the radios on the phone is used. PocketSearch is on average 23 times more energy efficient than 3G, 41 times more energy efficient than EDGE and 11 times more energy efficient that 802.11g.

Note that the gap in the energy efficiency between PocketSearch and the different radios on the phone is larger than the corresponding gap in the user response time shown in Figure 2.16. This happens because, as shown in Figure 2.18, PocketSearch conserves energy in two ways. First, no data are transmitted or received in the case of a cache hit, and thus the overall power consumption of the phone remains low ($900mW$ vs. $1500mW$). Second, since PocketSearch achieves a user response time that is an order of magnitude lower compared to when the radios on the phone are used (4 seconds vs. 40 seconds), the per query energy dissipation is significantly lower for PocketSearch.

## 2.6.2 Cache Hit Rate

To quantify the cache hit rate achieved by PocketSearch for a typical user, we used anonymized search query streams from the mobile search logs. To ensure a



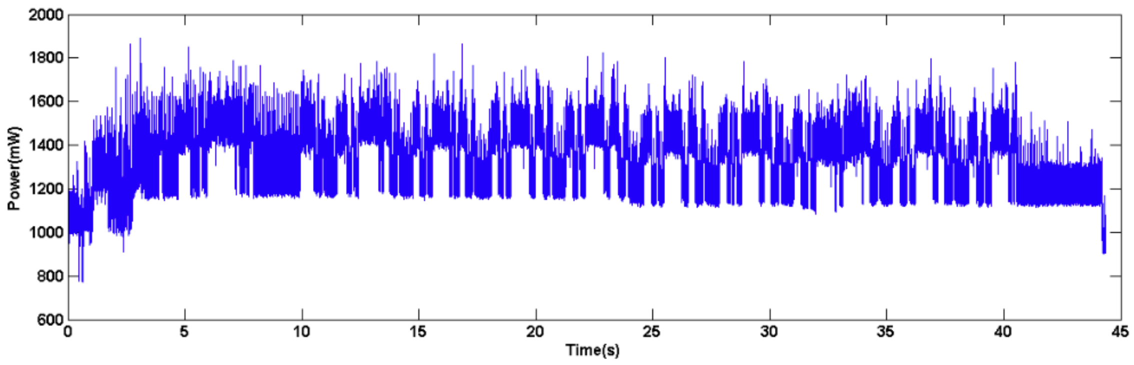


Figure 2.18: **Total time and power consumption for serving 10 consecutive queries through PocketSearch (top) and the 3G radio (bottom).**

representative and unbiased selection of search query streams, we classified users in four different classes based on their monthly query volume. Table 2.5 shows the different user classes and the percentage of users in the mobile search logs that belong to each class. Note that we ignore users that submit fewer than 20 queries per month for two reasons. First, PocketSearch targets users that frequently access the Internet and search the web. Second, as higher-end smartphones with advanced browsing capabilities become more and more available, the average monthly query volume submitted by individual users will increase beyond the threshold of 20 queries per month.

For the experiments described in this section, we randomly selected 100 anonymized users from each class shown in Table 2.5 and extracted their search query streams from the mobile search logs over a period of one month. Each of the 400 search query streams

Table 2.5: **Classes of users and their characteristics.**

| **User Class** | **Monthly Query Volume** | **% of Users** |
|---|---|---|
| **Low Volume** | [20,40) | 55% |
| **Medium Volume** | [40,140) | 36% |
| **High Volume** | [140,460) | 8% |
| **Extreme Volume** | [460,$\infty$) | 1% |

was replayed against the PocketSearch cache that was generated using the mobile search logs of the preceding month. The resulting cache contained approximately 2500 search results that corresponded to 55% of the cumulative *query-search result* volume in the search logs. Note that the data used to build the cache and the data used to extract the 400 query streams were non-overlapping.

### Hit Rate Results

Figure 2.19 shows the average hit rate for each user class described in Table 2.5. On average, 65% of the queries that an individual user submits are cache hits, and can be served 16 times faster. By examining Figure 2.19, it becomes apparent that the cache hit rate increases with the monthly query volume. PocketSearch achieves a cache hit rate of approximately 60% for the low volume class, which immediately jumps to 70% for the medium volume class and to 75% for the high and extreme volume classes.

Figure 2.19 also shows the average cache hit rate for every user class in the cases when PocketSearch uses only either the community or personalization component of the cache. When only the community component of the cache is used, new queries and search results selected by the user are not cached over time and, therefore, the cache cannot take advantage of the repeatability of mobile queries. When only the personalization component of the cache is used, the cache is initially empty and, therefore, cache hits are achieved only from repeated queries.

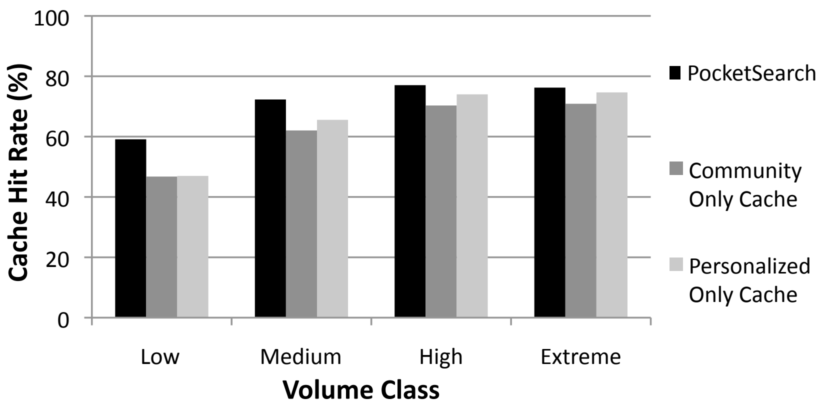
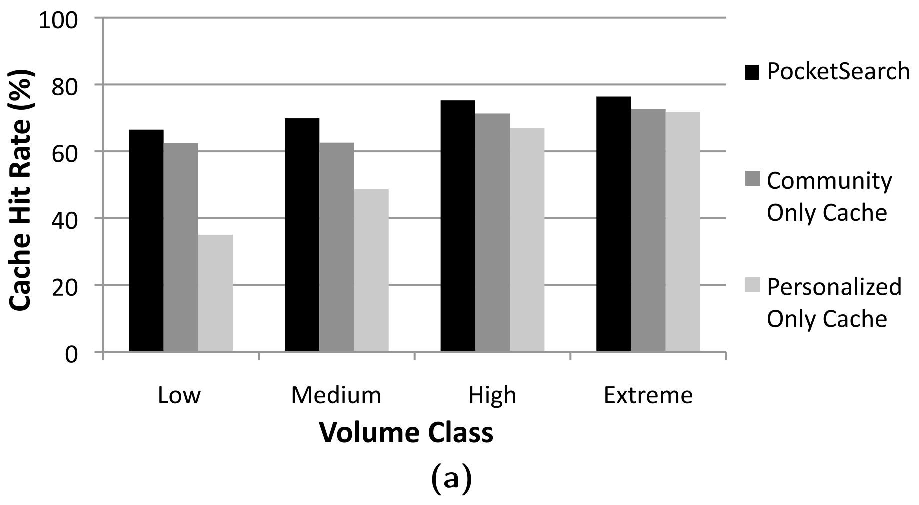


Figure 2.19: **PocketSearch's average cache hit rate.**

As Figure 2.19 illustrates, when only the community part of the cache is used, the average hit rate across all user classes is reduced from 65% to 55%. What is even more interesting is the fact that the hit rate seems to increase monotonically with the monthly query volume. Even though the exact same cache is used across all classes (since personalization is not used), the users that submit more queries seem to also experience higher hit rates.

When only the personalization part of the cache is used, the average hit rate across all user classes is reduced from 65% to 56.5%. Note that for every user class, the personalization part of the cache achieves the same or higher hit rate compared to the case where only the community part of the cache is used. This is another indication of the high repeatability of mobile queries that the personalization part of the cache is able to capture. In addition, the fact that the cache hit rate increases for users with higher query volumes, due to the personalized component of the cache, shows that users with higher query volumes repeat the same queries more often.

Even though users repeat mobile queries frequently, the community part of the cache is still very important for the overall user experience. Figure 2.20 shows the average hit rate for the different user classes during the first week [Figure 2.20(a)] and first two weeks [Figure 2.20(b)] of the one-month long query streams. Note that after



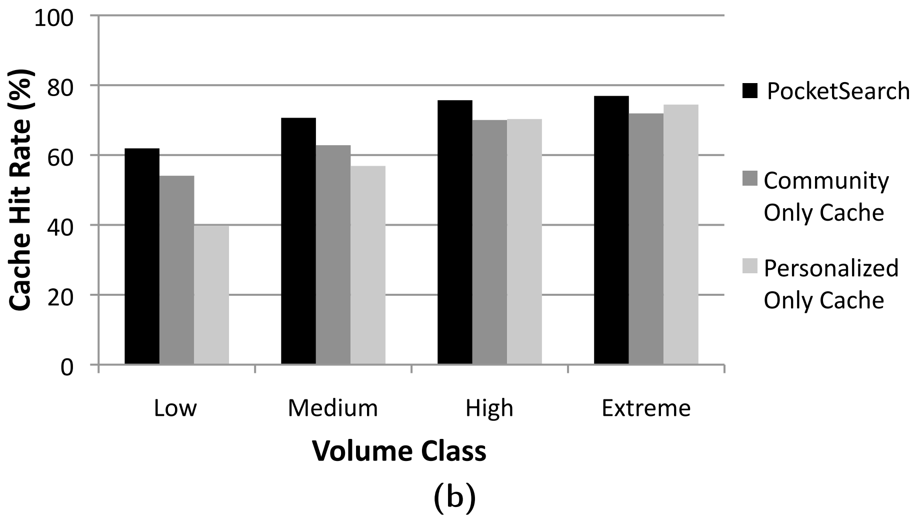


Figure 2.20: **Average cache hit rate across the four user classes for (a) the first week and (b) the first two weeks of the month.**

the first week, the hit rate of the personalization component of the cache remains lower than that of the community component of the cache. In particular, the fewer queries a user submits, the more time it takes the personalization component to warm up and be able to take advantage of the repeated queries. However, even during the first week, PocketSearch cache is able to provide the same hit rate with the one achieved in Figure 2.20 after a month. The community part of the cache provides a warm start for PocketSearch and good initial search user experience.

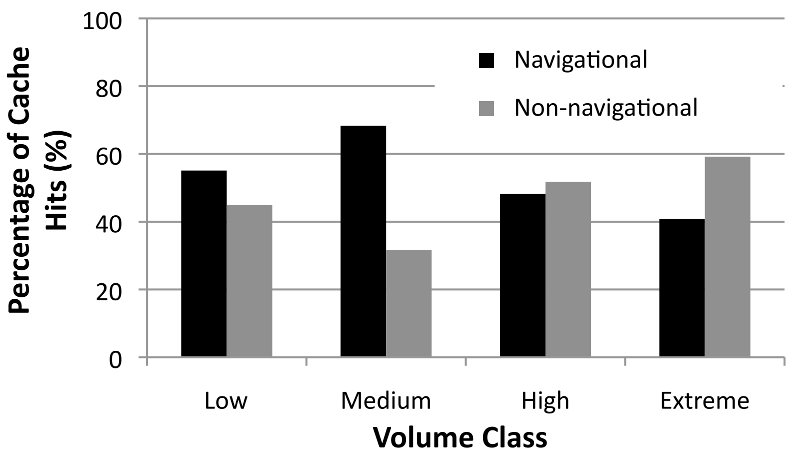


Figure 2.21: **Breakdown of PocketSearch's cache hits into navigational and non-navigational across the four user classes.**

The breakdown of the queries that result into a cache hit can be seen in Figure 2.21. On average and across all user classes, 59% of the cache hits are navigational queries[5] (*e.g.,* "facebook", "youtube", *etc.*) and the rest 41% are non-navigational queries. The non-navigational hit rates are significantly increased or even doubled when compared to the medium volume class for both the high and extreme volume classes. This trend indicates that higher volume users tend to submit more diversified queries. However, even for this type of users, PocketSearch is able to achieve high hit rates by taking advantage of the repeatability of mobile queries with its personalization component.

### Daily Cache Updates

To understand how changes on the set of popular queries and search results on the server affect PocketSearch's cache hit rate, we repeated the same experiments while updating the PocketSearch cache on a daily basis, as described in Section 2.5.4. On average, across all user classes, PocketSearch achieves a cache hit rate of 66% when daily updates are used. This incremental improvement of 1.5 percentage points (66%

[5]These are queries that current browser cache substring matching techniques could also serve. As more and more mobile services become available through dedicated applications, the volume of navigational queries may decrease but is expected to remain substantial. Some users still prefer to access the service by typing its name in the search text box.

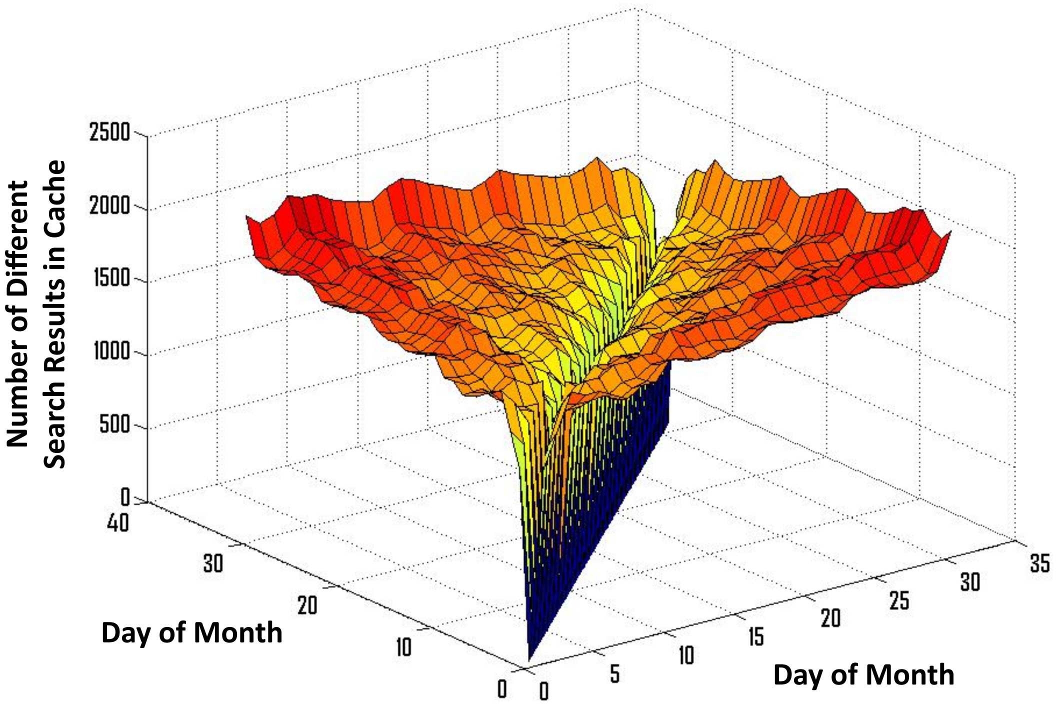


Figure 2.22: **Number of different search results in the community part of the cache for every pair of days over the one month period used for evaluation.**

vs. 65% hit rate when daily updates are not used) is due to the fact that the most popular set of queries and search results did not change significantly over the one month period we examined. As Figure 2.22 shows, on average, 40% to 50% of the approximately 2500 search results in the cache changed across days. This indicates that the search results that are responsible for the majority of hits in the community part of the cache are among the top 1000 most popular search results that always remain popular across different days.

### 2.6.3 Web Search Service Performance Benefits

The example of PocketSearch highlights the impact that the pocket cloudlet architecture can have on mobile user experience. Using approximately only 200 KB of DRAM

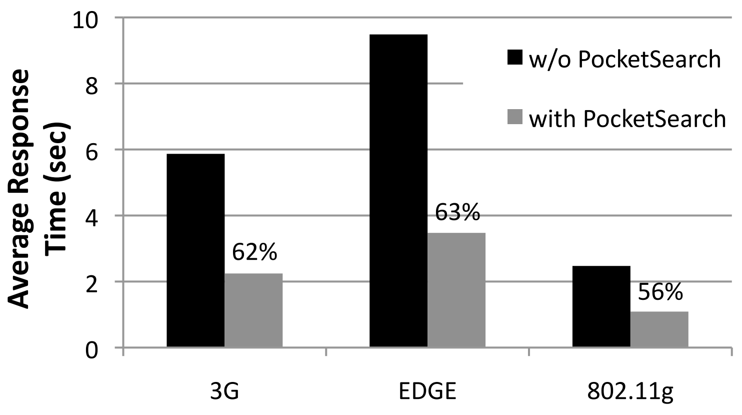


Figure 2.23: **Average user response time for the different communication interfaces with and without PocketSearch. The percentage by which PocketSearch reduces the average user response time is also shown on top of the bars.**

and 1MB of flash memory, PocketSearch can instantly serve, on average, 66% of the query volume that users submit. The benefits are manifold:

1. *User response time (latency)*: In 66% of the cases, PocketSearch can provide search results locally with an average response time of only 378msec. Figure 2.23 shows the average user response time for the various communication interfaces with and without PocketSearch. PocketSearch reduces the average response time by 56%-63%.
2. *Energy consumption*: Communication interfaces are among the most power-hungry components of mobile devices. In 66% of the cases in which PocketSearch can provide search results locally, the energy consumption per query is only $3.8J$. As Figure 2.24 shows, PocketSearch reduces the average energy consumption per query by 60%-64%.
3. *Communication bandwidth and data center load*: By serving 66% of the queries locally, PocketSearch reduces the communication bandwidth and the search engine data center load by the same percentage.

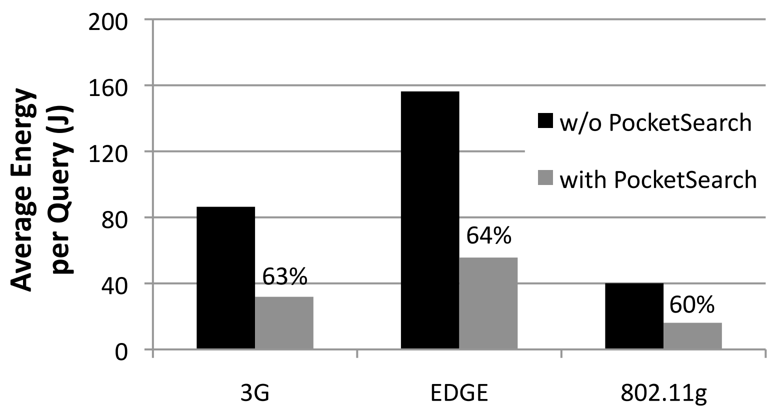


Figure 2.24: **Average energy consumed per query for the different communication interfaces with and without PocketSearch. The percentage by which PocketSearch reduces the average energy consumption is also shown on top of the bars.**

## 2.7 Architectural Considerations

Even though PocketSearch focuses on search services, the proposed architecture can serve as a template architecture for a broader family of cloudlets. Several other mobile cloud services beyond web search could leverage the same pocket cloudlet architecture, but each service eventually imposes its own memory requirements that might be very different when compared to PocketSearch. For instance, as Table 2.1 shows, a mapping service cloudlet would require approximately 25 GB to cache all the map tiles for the user's state. Furthermore, creating a yellow pages cloudlet requires storing information about 23 million businesses across the United States, which according to Table 2.1 corresponds to approximately 100 GB. Similarly, web content, mobile ads and other pocket cloudlets impose their own requirements.

Even though each pocket cloudlet might share the same architecture and design principles, when multiple cloudlets with different requirements run on the same device, they naturally compete for resources within themselves and with other user applications. Managing the system resources properly across multiple cloudlets poses several architectural challenges.

**User versus pocket cloudlets**: The operating system will need to limit memory consumption such that enough memory is available to user data and applications. The more content cloudlets cache, the larger their indexes become. These indexes are stored in main memory and compete with regular user application memory usage. Pushing the indexes (or part of them) into slow storage-class memory would significantly affect cloudlet performance. We suggest the adoption of an intermediate tier consisting of fast storage-class memory, as described in Section 2.3, to address this problem.

**Pocket cloudlet interactions**: Many pocket cloudlet services cache related data. For example, when the user performs a web search, both search and ad cloudlets are invoked for the same query. In addition, the results will point to related web content, map tiles and yellow page entries. Different cloudlets have distinct storage requirements, and their relative storage allocation should take this into account. In addition, we believe that when memory needs to be reclaimed and cache entries evicted, it should be done in a coordinated fashion. If a particular query misses in the local search cache, there is not much benefit in hitting the ad cache because the latency bottleneck to service this query will be waking up the radio. Cache eviction policies should be managed by the operating system and coordinated such that closely related items are evicted together.

**Security**: Some cloudlets may include sensitive user and/or application data in their caches. Consequently, other cloudlets should not be allowed unrestricted access to those cache contents. For example, a map cloudlet should not be allowed to access information regarding a user's recent bank transactions. We envision the operating system will provide such isolation and access control.

## 2.8 Related Work

This section compares the Pocket Cloudlets architecture with other proposed works that employ caching and other complimentary approaches for the purpose improving the user experience of cloud services. Our mobile search characterization is also compared with previous work.

### 2.8.1 Caching of Services

To improve user experience, caching of information on client devices has been applied in the past in the context of web content [65, 97, 103, 113, 115] and advertisements [50]. Our work on Pocket Cloudlets differs in three ways. First, we describe a generalized mobile caching architecture that can be applied to various cloud services. Second, PocketSearch is complementary to these efforts, in the sense that it focuses on caching search results and not web content. Third, as opposed to the previous work, PocketSearch was designed and implemented on an actual smartphone, addressing the challenges of building such an architecture on a mobile device.

File server caching systems, such as Coda [18], have focused on remote/offline access of files. Such schemes do not distinguish between a search result and a standard webpage HTML file, let alone allow for personalized ranking of results within a page. They focus on the level of files as opposed to individual service data items. Since these systems have no notion of search results, they do not take into account search trends into designing optimal client-side caching strategies. For instance, our data-driven approach indicated that by storing individual search results, storage requirements can be reduced by a factor of 8. In addition, PocketSearch combines both personal and community access characteristics to decide what search results to cache.

Approaches focusing on client side database caching have been combined with web browser caches to locally serve dynamic pages [13, 41]. Such approaches, however, aim to reproduce the exact same dynamic page as the servers (*e.g.*, search engines). They do not focus on assessing the popularity and cacheability of individual objects (*e.g.*, search results) within the dynamic page (*e.g.*, search results page), let alone personalize their ranking.

Content delivery or distribution networks (CDNs), such as the commercially available Akamai (`www.akamai.com`), maintain geographically distributed data caches aiming to minimize user access time to content. CDNs are complementary to PocketSearch. CDNs have great impact on wired networks where the latency between the client device and the nearby CDN node is low. However, on mobile devices, the bottleneck is the wireless link. By appropriately combining Pocket Cloudlets and CDNs, distributed cloud services that provide instant user experience can be created.

In the desktop search domain, the temporal locality and repeatability of queries across users has been exploited by proposing a server-side search caching scheme to reduce the load of search engines and improve user response time [36, 96, 153]. In the mobile domain though, the performance bottleneck is not the search engine but the slow radio link. As opposed to these approaches, PocketSearch is a client-based search cache that lives on the phone and enables search results to be displayed instantaneously as the user enters a query.

### 2.8.2 Other Approaches to Improve User Experience: Keyword Auto-completion

Previous work has proposed keyword auto-completion services on mobile devices [71] in order to facilitate and speed up the entry of query strings by the user. Such services are already popular on smartphone browsers and other mobile search applications and are complementary to PocketSearch, which focuses on providing fast search results and not query suggestions.

Browsers and dedicated search applications on high-end smartphones, such as Android phones and the iPhone, have recently enabled website suggestions as the user types a query. This is done in two ways.

First, for every new letter typed in the search box, a query is submitted in the background to the server for the partially entered query term and the most popular search result is returned as a website suggestion to the user. Note, that in this case, a regular search query has to be submitted to the server over the radio link and, therefore, the usual slow mobile search experience takes place.

Second, as the user types a query, a substring matching algorithm between the partial query string and all the website addresses in browser's cache can instantly provide the most relevant website that has been previously visited by the user. Unfortunately, this approach only works for a portion of the navigational queries.

### 2.8.3 Search Log Analysis

There have been several research efforts on understanding mobile search behavior through search log analysis [27, 28, 68, 69, 70, 71, 72, 155]. These efforts have analyzed search query volumes that vary from hundreds of thousands to several tens of millions of queries. The search log analysis presented in this paper differs in two fundamental ways. First, we analyze 200 million mobile search queries, a query volume that is one order of magnitude larger than the query volume used in any other mobile search log study; volumes that large have been studied before only for desktop search [133]. Second, besides reporting similar observations on the locality of queries across mobile users, we also study in detail the repeatability of mobile queries for individual users.

## 2.9 Conclusions

To reduce the dependence of non-geo-locality services on long-range communications, this chapter presented the Pocket Cloudlets architecture. A pocket cloudlet is a mobile device-resident architecture that selectively caches parts of a cloud service and serves requests locally, when possible. By serving requests locally, pocket cloudlets help reduce the need for long-range communications, resulting in improved user experience.

The Pocket Cloudlets architecture is a collaborative and adaptive caching scheme. Locally constructed personal user models are collaboratively combined to construct community models. Both personal and community models are then used to decide on selective caching, ranking and adaptive prefetching of service data items.

To demonstrate the potential of the Pocket Cloudlets architecture, this chapter presented PocketSearch, a mobile search pocket cloudlet. To guide the design of PocketSearch, we studied the cacheability of mobile search by studying both community and personal user access patterns. We found that mobile search is significantly more concentrated, as compared to desktop search, on both a community and a personal user basis. By utilizing both personal user and collaboratively-generated community access models to maximize its hit rate, we showed that PocketSearch can serve 66% of the web search queries without having to use the slow 3G link and by using only a small fraction of the mobile device storage resources. As a result, PocketSearch leads to significantly reduced cellular communication bandwidth (66%), user response time (62%) and power consumption (63%).

The example of PocketSearch demonstrates that collaborative and adaptive pocket cloudlets can efficiently augment traditional cloud services. Furthermore, as our analysis shows, the NVM storage capacity of modern smartphones is already high enough to support a rich set of traditional cloud services besides web search. By augmenting existing cloud services with a mobile-device resident pocket cloudlets architecture, a significant percentage of cloud service requests may be served locally, mitigating pressure on both cellular links and data centers. Above all, by avoiding, when possible, the use of slow and costly long-range cellular communications, the mobile user experience improves significantly.

# Chapter 3

# RegReS Middleware for Geo-locality Services

The previous chapter presented the Pocket Cloudlets architecture that reduces the use of slow and costly long-range communications for non-geo-locality cloud services. For geo-locality services, to completely alleviate their dependence on long-range communications, this chapter presents the Region-Resident Services (RegReS) middleware. RegReS enables the full hosting of such services on regional confederations of mobile devices that collaborate over short-range communications. To avoid wasting the resources of mobile devices when running resource-intensive tasks (*e.g.,* SignalGuru), RegReS does not use all available devices but seeks to maintain a specified service carrier density. To maintain the specified density in highly-volatile mobile networks, RegReS uses a fully-distributed, collaborative and adaptive scheme to estimate its current value and guide service spawning decisions. We explore the ability of RegReS to estimate and maintain the specified density of service carriers across a wide range of configurations and study the effects of different service spawning policies and criteria. Furthermore, we examine which dynamic factors may influence RegReS's performance, presenting and evaluating a mechanism to adapt.

## 3.1 Introduction

Networked mobile computing devices are becoming increasingly pervasive. Powerful smartphones are nearly ubiquitous. Furthermore, soon a large percentage of vehicles will be equipped with advanced Personal Navigation Devices (PND) that will have not only motion sensors (GPS, accelerometer, gyroscope, compass) but also fast, short-range ad hoc communications (*e.g.,* DSRC [148]).

These trends are prompting exciting location-based services that leverage opportunistic local sensing. For instance, on-board mobile devices using their wireless interfaces, GPS, gyro and accelerometer information can help estimate traffic conditions [99], detect road abnormalities [33], collect information for available parking spots [19, 98], predict the schedule of traffic signals [85] and measure air or noise pollution [116].

Typically, location-based services have the data geo-locality property. This means that the data of the location-based service is both generated (sensed) and consumed locally, *i.e.,* within a defined geographic region. Devices need to be within a given geographic region to sense the necessary data. Also, only devices within this geographic region are interested in consuming this sensed data. For example, in the case of a parking spot availability service for Princeton, only vehicles moving within Princeton can detect the parking spot availabilities as they drive by [98] and only vehicles in Princeton are interested in finding available parking spots there, *i.e.,* in their vicinity. A driver would be willing to drive at most a couple miles in order to find a spot to park.

Traditionally, location-based mobile services have been architected atop Internet cloud servers accessed via long-range cellular communications. However, as noted in Section 1.2, long-range communications are scarce, costly and lead to poor user experience. Long-range communications should thus be used judiciously and only when necessary.

In this chapter, given the data geo-locality property of many location-based services and the limitations of the traditional mobile cloud service architecture, we explore RegReS. RegReS enables the full hosting of geo-locality services on the mobile devices in the region of interest, with the devices collaborating over short-range communications to form a distributed computing platform, obviating the need for long-range communications and cloud server infrastructure. Given the high node density, mobility and availability of free device-to-device (ad hoc) communications, such a grassroots distributed computing platform is particularly suited for geo-locality services, as highlighted previously in [19, 89].

While mobile devices are ever more powerful, their resources are still often constrained and should thus be used judiciously. As we show in the next chapter, some services, like our SignalGuru, may be particularly resource-intensive and hog certain mobile device resources (*e.g.,* CPU). As a result, services should often be spawned on only as many mobile devices as necessary.

RegReS thus allows each service to specify its desired service carrier density (the number of mobile devices that should host this service within a region) along with its region of interest and lifetime. The RegReS middleware, which runs on the carriers, ensures that this target carrier density is maintained in a distributed fashion. RegReS uses a *collaborative* and *adaptive* estimation scheme to track and estimate the current carrier density for a service. RegReS then employs spawn policies and carrier selection criteria to decide when and which nodes to spawn as new carriers.

Figure 3.1 illustrates an instance of a RegReS-enabled service. As shown here, the region of a service is a polygon and contains three types of nodes (mobile devices). The nodes that carry the service are termed *service carriers.* The nodes that do not carry the service but have the RegReS middleware and could thus potentially become future carriers are termed *RegReS nodes* or just *nodes.* Last, the non-RegReS nodes that do not have the RegReS middleware cannot communicate (collaborate) with

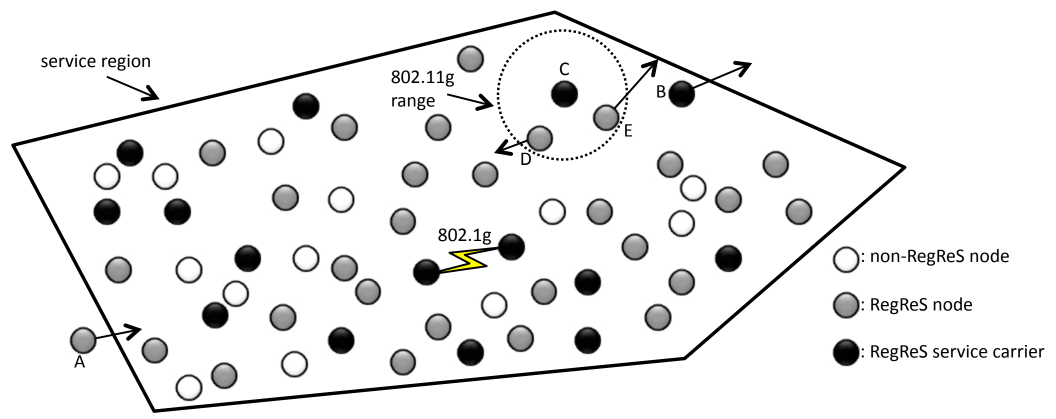


Figure 3.1: **RegReS maintains each service within the specified region (polygon) and for the specified service lifetime.**

other regional nodes for the maintenance of a service and are thus ignored in our discussion.

In this chapter, the contributions of our work on RegReS are as follows:

1. We identify the five traits that middleware that support grassroots geo-locality services possess and argue for service carrier density as the suitable metric for such mobile services. Using this metric, our solution is the first to account for all five traits.
2. We propose the first collaborative approach in Mobile Ad hoc Networks (MANETs) that *adapts* to environmental parameters and maintains a targeted density of services, and we show its effectiveness on a testbed.
3. We demonstrate RegReS's potential by using it as the middleware for the deployment of a collaborative parking availability service.

The structure of this chapter is as follows. Section 3.2 surveys related work and Section 3.3 describes the design of RegReS. Next, Section 3.4 describes the evaluation methodology, and Section 3.5 presents performance results.

## 3.2 Background and Related Work

Typically, for grassroots geo-locality services, a certain number of service carriers must be maintained in the region of interest for sensing, storage and computation. We term this *service carrier density (d)*. We argue that each application service should choose and specify its own desired $d$ in the targeted geographical region of deployment so as to be able to reflect its own cost-performance tradeoffs.

For instance, in an application that estimates average vehicle speed on roads in a region, sensing may be noisy, as a given vehicle that reports its speed may be moving faster or slower than most other vehicles. With a sufficient carrier density across the region of interest, a good enough sampling density can be attained, with outliers removed. On the other hand, having too many vehicles as service carriers leads to high cost in terms of computing, storage and communications overhead.

Similarly, for a parking availability service that detects free parking spots with noisy ultrasonic sensors [98], higher density of service carriers leads to more frequent road scanning and more robust detection. Yet, too many carriers lead to unnecessary consumption of regional computational resources.

While most geo-locality services need to maintain such a desired regional sensing capacity based on their own cost-performance tradeoffs, there is often no intuitive definition of sensing range. For example, in the case of traffic estimation, what is the sensing range for a vehicle that is reporting its own vehicle speed measurement?

On the other hand, proposed geo-locality services do not typically need hard guarantees on sensing coverage (*k-coverage*) as they often do not focus on life-critical applications (*e.g.,* intrusion detection in sensor networks). Opportunistic sensing is often both sufficient and the only approach possible given the uncontrolled mobile device (*e.g.,* smartphone, vehicular computer) density and mobility.

Therefore, we target service carrier density as the metric for grassroots geo-locality services. RegReS allows services to specify their desired carrier density ($d$) along with their region and lifetime. This carrier density determines the population of carriers that RegReS should seek to maintain within the service's region and for the specified service's lifetime.

### 3.2.1 Traits of Grassroots Geo-locality Services

Grassroots geo-locality services possess five key characteristics, which make the maintenance of the target number of service carriers a challenging task. As Table 3.1 shows, no prior works sufficiently tackle all five challenges:

1. *Variable (high) node mobility:* Nodes (mobile devices) are carried by people as they walk, drive or ride the public transit. As a result, mobility varies greatly. Schemes need to adapt to variable node mobility.
2. *Uncontrolled node mobility:* Proposed schemes should not assume that the movement of nodes can be orchestrated to better support services. Schemes need to be able to leverage nodes opportunistically.
3. *Uncontrolled and variable node density:* The node density of opportunistic mobile networks varies and can be especially high in cities, up to several hundreds or thousands in a $5km^2$ region [121]. More often than not, only a small fraction of the nodes is necessary to provide a given service. Schemes need to adapt and selectively use only as many nodes as needed.
4. *Significant service activation/replication (spawn) cost:* Geo-locality services can sense and gather data rapidly, leading to a large amount of state that needs to be transferred whenever a new carrier is spawned. Furthermore, there is no control over the software each mobile device has. As a result, signed code modules may need to be moved to new carriers as well. Spawns can, hence, be several tens of KB. Proposed schemes should thus retain a service carrier as long as possible (while it remains in the region), and minimize the spawning of new carriers.

Table 3.1: **Comparison of RegReS with related work. No prior work adapts sufficiently to system dynamics (*e.g.*, variable and uncontrolled node mobility, density, *etc.*).**

| | **service replication** | | | | | | | | **estimation schemes** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **epidemic push** | **epidemic pull** | **probabilistic spawning** | **N initial replicas** | | | **k-coverage** | | **centralized** | **distributed** | **RegReS** |
| | [19, 89, 92, 93, 95] | [91, 95] | [88, 102, 157] | [77, 87] | [12, 137] | [91] | [75] | [16, 138] | [119] | [38, 42, 89, 156] | [84] |
| **high mobility** | √ | √ | √ | X | √ | √ | X | √ | √ | √ | √ |
| **uncontrolled mobility** | √ | √ | √ | X | √ | √ | X | X | √ | √ | √ |
| **adaptivity** | X | X | X | X | X | X | X | X | X | X | √ |
| **spawn cost** | X | √ | √ | X | X | √ | X | X | N/A | N/A | √ |
| **fault tolerance** | √ | √ | √ | X | X | X | √ | √ | X | √ | √ |

5. *Challenging operating environments:* Proposed schemes need to be robust to node failures; nodes may crash, run out of battery, get powered off by their owners, or just exit the region of interest.

### 3.2.2 Service Replication Literature

Proposed grassroots approaches targeting geo-locality services [19, 89, 95] have been largely based on schemes that epidemically *push* the service on *all* available nodes. However, as noted earlier, particularly in dense urban environments, using all nodes for a given service is both unnecessary and wasteful.

Other approaches, in contrast, do not proactively push the service to all regional nodes, but have the nodes interested in the service epidemically *pull* it [95] or *subscribe* to receive it [91]. However, such approaches cannot guarantee that the critical number of service carriers will be available; often a disparity will exist between the number of service consumers/subscribers and the number of carriers that are necessary to support the service.

A third class of approaches tries to control the number of service carriers by instructing current carriers to epidemically spawn new carriers with a defined probability [88, 102, 157]. While such approaches are not as wasteful as approaches that use all nodes, our results in (Section 3.5.1) show that a fixed spawn probability may work well for a specific case, but fails to adapt across different configurations (node mobility, region size, *etc.*).

A fourth class of theoretical works [12, 77, 87, 91, 137] begin by creating the desired number of service replicas/tokens and assume that the services can be moved between nodes *without accounting for node failures.* Lastly, *k-coverage* literature in sensor networks assumes either that nodes are *static* [75] or that their mobility can be *controlled* [16, 138]. None of these assumptions hold in real-world opportunistic networks. In such networks, nodes may be highly mobile and move in a self-determined fashion, *e.g.,* mobile phones carried by users as they drive to their personal destination.

### 3.2.3 Density Estimation Literature

In RegReS, service carriers track their density across the region so that they can make informed decisions about whether to spawn a new carrier or not. No prior art has used service density as the metric to guide service activation/replication. Furthermore, despite the importance of determining density in MANETs, little work has been done so far in estimating it. In [119], the authors propose a *centralized* node census approach, which is not suited for highly-distributed opportunistic networks.

Our collaborative distributed density estimation scheme draws from [156]. In contrast to [156], RegReS carriers do not exchange complete logs of raw measurements with their timestamps, but only the estimates they themselves have built or received from other carriers together with the confidence value for these estimates. Such estimates may be based on multiple such measurements. Thus, the amount of information exchanged for density estimation is significantly reduced in RegReS.

Collaborative schemes for estimating local density are also used in [38, 42, 89] to guide (request or vehicle) routing decisions. However, these schemes focus only on local density and limit the information exchange to only between direct neighbors within communications range and do not leverage opportunistic forwarding.

Above all, RegReS, unlike prior art, adapts its estimation scheme to node dynamics. As shown in Section 3.5.1, this adaptivity is of critical importance.

### 3.2.4 RegReS for Geo-locality Services

Here, we outline how RegReS handles the five key traits of grassroots geo-locality services, tackling a key gap that has not been addressed by prior research:

1. *Uncontrolled and variable node mobility:* RegReS uses nodes opportunistically and adapts its density estimation scheme to their mobility patterns.
2. *Uncontrolled and variable node density:* RegReS targets a service-specified density of regional nodes (the designated carriers) for maintaining the service.
3. *Significant service activation/replication (spawn) cost:* RegReS uses nodes as service carriers for as long as they remain in the region, as opposed to *k-coverage* schemes that use sleep-schedule-based activation [16, 138]. Through simulations, we found that this can result in up to $4.9\times$ reduction in the number of spawns for the experimental scenarios considered here.
4. *Challenging operating environments:* RegReS is fully distributed and adaptively tracks and reacts to the current density of carriers. This makes it robust to carrier failures and departures from the region.

### 3.2.5 Privacy, Security and Trust

Opportunistic and collaborative services, like the ones that RegReS enables, come with privacy, security and trust implications. RegReS can use DLT certificates [90] or a TPM [131] to ensure trust in the service carrier operations. Furthermore, spatio-temporal cloaking [49] and other proposed approaches for grassroots participatory sensing [45, 56] can be used in RegReS to provide security and privacy.

## 3.3 RegReS Design: Carrier Density Maintenance

To maintain the desired carrier density, RegReS uses a collaborative and adaptive distributed approach to *estimate* it across the region and react accordingly. It determines *when to spawn* new carriers and *how to select the carriers*[1]. The design of RegReS is thus broken down into three sub-problems:

[1]The maintenance of the service is the responsibility of only the existing carriers. Non-carrier nodes are not involved and, hence, do not incur any overhead.

**1. How to estimate carrier density:** Carriers measure the density within their communication range periodically. What information should carriers exchange based on their measurements and how should it be used to calculate an estimate? Above all, how can this estimation scheme automatically adapt to system parameters (node mobility, region size, *etc.*)?

**2. When to spawn:** When can a carrier be sufficiently confident, given its estimate, that it needs to spawn a new carrier?

**3. How to select carriers:** Once a carrier decides to spawn a new carrier, which node should it pick?

Solutions to these three sub-problems constitute the novel contributions of RegReS. Other peripheral functions, such as service discovery and updates, are done epidemically.

### 3.3.1 Carrier Density Estimation

RegReS estimates carrier density in a collaborative fashion, through small metadata packets that nodes broadcast every $P$ seconds. These Service Advertisement (SA) packets contain information about the services (if any) nodes carry. They allow for: 1) service discovery, 2) service version updates, and 3) discovery of potential carrier nodes if spawning is needed. RegReS leverages these packets to measure local density as well as allow carriers of the same service to exchange density estimations. The exact format of SA packets is shown in Figure 3.2.

The carrier density $d$ is expressed in units of number of carriers per $\pi R^2$ area, where $R$ is the communication range:

$$d = \frac{N_{carriers}}{\alpha \times Area} \ , \quad \text{where} \quad \alpha = \frac{1}{\pi R^2} \tag{3.1}$$

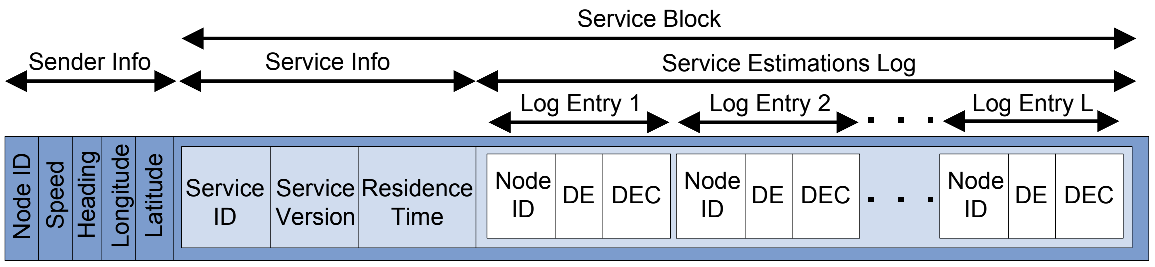


Figure 3.2: **Service Advertisement packet.** *Node IDs* **are unique identifiers of nodes.** *<Speed, Heading, Longitude, Latitude>* **determines the average speed, direction that a node is moving towards and current location, and are used in carrier selection to calculate ERRT.** *Service ID* **and** *Version* **uniquely identify each carried service.** *Residence Time* **tracks the number of periods that the carrier has resided within the region of this service and is used for determining the value of $f$. The** *<Node ID, DE, DEC>* **triplets are used for collaborative density estimation. One such** *Service Block* **is included in the SA packet for each service the node is carrying.**

If, for example, $N_{carriers} = 100$ is the target population of carriers within the service's region, $Area = 2216m \times 2216m$ is the region size and $R = 250m$, then in RegReS, the targeted service carrier density is $d = 4$ carriers per communication range.

### Estimation Algorithm

Every $P$ seconds[2], carriers measure the number of other carriers (of the same service) in their range, by listening to SA packets. Measurements are exponentially-weighted-averaged to form a density estimate that is biased towards newer measurements. The factor $0 \leq f < 1$ by which the measurements' weights decay over time is called the *decay factor*. The smaller $f$ is, the faster measurements decay. If $m_{-k}$ is the measurement done $k$ periods ago, the current Density Estimate (DE) is calculated as follows:

$$DE = \frac{\sum_{i=0}^{k}(1-f) \times f^i \times m_{-i}}{\sum_{i=0}^{k}(1-f) \times f^i} \tag{3.2}$$

[2]Nodes are time-synchronized with a GPS device. They use a random backoff scheme to broadcast their SA packet within each period.

This formula can be rewritten in the form of a moving average to avoid maintaining the full measurement history:

$$DE = f \times DE_{old} + (1 - f) \times m_0 \tag{3.3}$$

The sum of the weights of these measurements converges to 1 and we term this Density Estimation Confidence (DEC). DEC is the *confidence* that a carrier has in its estimate. It grows over time as the carrier gathers more and more measurements:

$$DEC = \sum_{i=0}^{k} (1 - f) \times f^i \rightarrow 1 \tag{3.4}$$

RegReS adapts the value of the decay factor $f$ to system parameters (node mobility, region size, *etc.*). The adaptation is based on a regression model described in Section 3.3.4. This adaptation influences both the value of $DE$ (Equation 3.3) and the rate with which carriers accumulate confidence (Equation 3.4) for their density estimates. As shown in Section 3.5.1, adaptation is critical and offers RegReS improved performance over a wide range of system configurations.

Carriers use SA packets to exchange DEs. SAs include a list of triplets <Node ID, DE, DEC> that record along with the estimate (DE), its confidence (DEC) and also the ID of the carrier node that had generated it (to detect and discard duplicates). Carriers for each service maintain and exchange a log of size $L$ of such entries. This exchange helps carriers populate their logs with triplets from other carriers and forms the basis of the collaborative density estimation scheme. A carrier uses the information in this log (that includes its own estimate too) to build a more accurate estimate by weighting the estimations (DE) using their confidences (DEC):

$$DE_{merged} = \frac{\sum_{i=1}^{L} DE_i \times DEC_i}{\sum_{i=1}^{L} DEC_i} \tag{3.5}$$

As new triplets are received, only the $L$ most confident ones are preserved. Log entries are decayed at the end of each period, by multiplying their DEC by the decay factor $f$. In this way, their effect in the averaging operation (Equation 3.5) is also decayed to reflect their increasing staleness.

### 3.3.2 Spawn Policies: When to Spawn?

The service carriers are mobile and stay within the service's region only for a limited amount of time. In order to maintain the desired carrier density, new carriers need to be spawned over time to replace carriers that exit. We propose and investigate three alternative spawn policies:

**Policy 1 (P1):** Spawn if $m_0 < d$. A carrier will spawn a new carrier whenever the measurement it made over the last period indicates that the existing carrier density is lower than the target value. Since carriers are highly mobile, their spatial distribution changes all the time and several transient carrier clusters and dispersals are created across the region. As a result, this spontaneous policy may end up overspawning carriers.

**Policy 2 (P2):** Spawn if $DE_{merged} < d$. A carrier bases its spawn decision on the merged density estimate it builds over time and not solely on the last measurement. Regardless of how long the carrier has resided within the region, it will spawn a new carrier whenever the value of its merged density estimate is less than the target density of carriers.

**Policy 3 (P3):** Spawn if $DE_{merged} < d \text{ and } DEC \geq C_{thres}$. The very first estimations that a carrier makes are not that accurate, as they are based on a limited number of measurements and exchanges (if any) with other carriers. It takes time for a carrier to build a more accurate and confident $DE_{merged}$. Therefore, a further optimization enforces a confidence threshold ($C_{thres}$). A carrier will spawn only if its DEC (as defined in Equation 3.4) exceeds the $C_{thres}$ threshold.

The confidence of a carrier's merged density estimation ($DE_{merged}$) could also be calculated based on the information in its log as $DEC_{merged} = \sum_{i=1}^{L} DEC_i$. This may seem like a reasonable scheme, but, in reality, fails to preserve the service within the region. The lower the existing carrier density (as a result of multiple carrier exits), the more existing carriers need to spawn new carriers, but the lower the density, the harder it is for existing carriers to encounter other carriers to populate their log with estimation triplets and, hence, build a high enough merged confidence that will allow them to spawn. Therefore, this confidence calculation scheme does not allow carriers to build enough confidence when they need it the most. As a result, the confidence of a carrier's $DE_{merged}$ is defined as in Equation 3.4 and grows as the carrier spends more and more time in the region. Confidence grows sublinearly, though, as old measurements are not as indicative as new ones. The rate at which confidence increases depends also on the value of the decay factor $f$.

Spawn packets are unicast UDP packets that contain the service ID, version, region, lifetime and data. The service data consist of service state as well as signed code modules, should the newly spawned carrier not have the necessary modules to run the service. The polygon that defines the geographic region of the service is described by a set of <latitude, longitude> pairs.

### 3.3.3 Carrier Selection Criteria: On Whom to Spawn?

Given the high speed of vehicles, the number of distinct carriers needed to preserve the desired density can be in the order of several thousands in our experimental scenarios. Spawns can be reduced by accounting for node mobility and selecting as carriers only nodes that will be staying within the region for more than a threshold amount of time.

The Estimated Residual Residence Time (**ERRT**) of a node, *i.e.,* the estimated amount of time left for which the node will be remaining within the region of the specific service, depends on its current location, heading and speed. Carriers calculate the ERRT of encountered nodes using the *<Speed, Heading, Longitude, Latitude>* information of the SA packets they receive. At the same time, carriers also calculate the mean of these ERRTs across nodes; this is termed Nodes' Mean Residual Residence Time (**NMRRT**). The NMRRT is calculated over a time window. Only ERRTs from SA packets sniffed over the last 10 cycles are included in the averaging process. In this way, values that are very stale are ignored.

Our carrier selection criteria select as carriers only nodes whose ERRT is greater than NMRTT by some factor. In Figure 3.1, for example, carrier C would only spawn the service on node D, if necessary. Node E will be leaving the region very soon and thus does not qualify to become a service carrier. Different carrier selection criteria may have varying effects on the uniformness of the carrier distribution across the region and we investigate this in Section 3.5.3.

### 3.3.4 Decay Factor Adaptation Model

A poorly chosen value for the decay factor can greatly impact the accuracy of the collaborative density estimation scheme and thus the accurate maintenance of the target density of service carriers. As Section 3.5.1 shows, a bad decay factor value may lead to errors as high as 41%. In more extreme scenarios (*e.g.,* of even higher or lower node speeds), the error could become even higher. It thus becomes critical to understand how the decay factor influences the density estimation scheme and, subsequently, how its value should adapt to dynamic system factors.

The fact that the decay factor $f$ greatly affects the performance of RegReS is also evident from Equations 3.3-3.5. More specifically, a small decay factor yields density estimates that are highly biased towards recent measurements while larger decay factors place more emphasis on older measurements. Intuitively, when the rate of change of carrier density (due to carrier exits and new carrier spawns) is high, newer measurements are a lot more indicative of the actual current density compared to older ones and thus the decay factor should be smaller. Conversely, when the rate of change of carrier density is low, older measurements reflect the current density of carriers almost as well as the most recent ones, and the decay factor should be larger.

Opportunistic networks, in general, and vehicular networks, in particular, are highly dynamic and, hence, a model is needed for automatically adapting the decay factor to system dynamics. As Section 3.5.1 shows, an accurate carrier density can be maintained only if the decay factor adapts to node mobility or region size changes[3]. Intuitively, these two factors determine the rate of changes as the faster nodes move or the smaller the region size, the faster existing carriers exit the region and new carriers need to get spawned. Conversely, for small speeds (or big regions), the rate of changes is lower.

As the basis of our adaptation model, and in order to track the rate of change, we choose the Carrier Mean Residence Time (**CMRT**) metric. The CMRT is the average amount of time that carriers reside within the region and we use it to derive a regression-based model for $f$. Carriers estimate and update the value of CMRT using the information in the SA packets received from other carriers. The carrier (past) Residence Time, as included in the SA packets, is added to the calculated (future) ERRT time to estimate the total residence time for each encountered carrier. The total residence times of the carriers are then averaged using a simple arithmetic mean over a time window to form CMRT. Only values from SA packets sniffed over the last 10 cycles are included in the averaging process. Armed with CMRT, nodes then use the regression model to select the decay factor $f$ they should use for DE and DEC calculations.

[3]We found that other system parameters like regional node population (Figure 3.4) or target density of carriers (Figure 3.7) do not affect the performance of the estimation scheme.

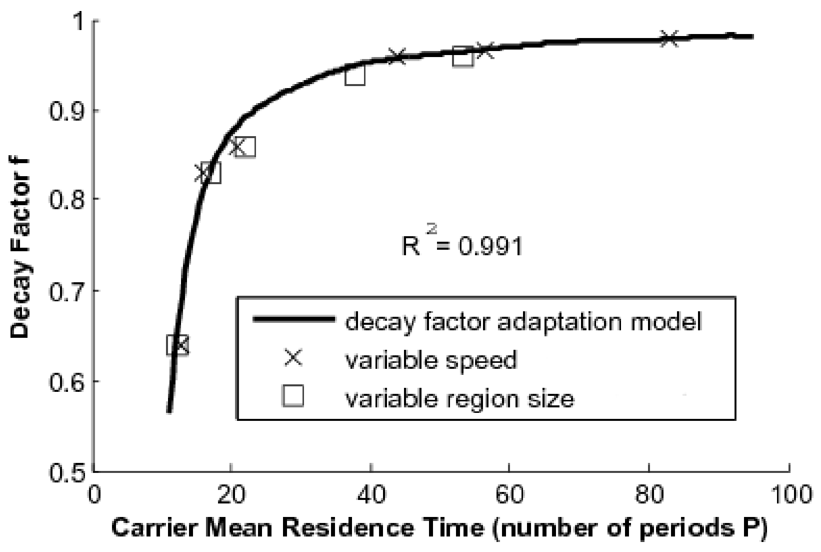


Figure 3.3: **Regression model for adaptation of decay factor $f$ to different system configurations. The data points for different node speeds and service region sizes correspond to the configurations of Figures 3.5 and 3.6, respectively.**

In Figure 3.3, we plot the values of the best[4] decay factor against CMRT for the configurations of variable speed of Figure 3.5 and the configurations of variable region size of Figure 3.6. Figure 3.3 also shows the regression-based approximation model that RegReS uses to adapt its density estimation scheme as a function of CMRT. The adaptation model is based on 9-th order polynomial regression. We found that higher orders or non-polynomial kernels do not improve the approximation accuracy significantly. Therefore, using the approximation of Figure 3.3, the best value for the decay factor can be determined for arbitrary configurations by only knowing the CMRT.

While our regression model is developed based on Random Waypoint (RWP) mobility model, our results in Section 3.5.1 show that the model is general and works effectively with real bus traffic traces as well.

[4]Ideally, the carrier density should always equal the target value. Thus, the error metric that RegReS seeks to minimize is the density mean absolute error calculated as the absolute differences between the actual density of carriers and the target density at each period, averaged over the duration of the experiments (360 periods). The best decay factor is the one that minimizes this error. An alternative error metric would be to penalize negative errors (actual density less than target) more than positive ones (actual density more than the target).

## 3.4 Methodology

This section describes the testbed and the mobility models that we used for the evaluation of RegReS.

### 3.4.1 Testbed

For the evaluation of RegReS , we prototyped a real system on the ORBIT[5] radio grid testbed [118] that provides a facility of 400 wireless Debian 4.1 nodes. For our experiments, we used up to 350 of these nodes configuring their Atheros AR5002X Mini PCI cards in 802.11a ad-hoc demo mode. The radio range was $250m$ and mobility was emulated by filtering out packets from nodes whose virtual distance was greater than this. Table 3.2 shows the default parameter values. Packet loss rate was in the range of 1-4%.

### 3.4.2 Mobility Models

We used two mobility models to evaluate the performance of RegReS:

1. *Random Waypoint (RWP):* The entry point of the nodes into the region is uniformly at random chosen on the border of the region. RWP then determines the travel path of the nodes. Node speeds are uniformly distributed in the range of $5m/s$ to $15m/s$ to emulate vehicular traffic. When a node crosses the region's border, the node is considered to have exited and thus removes the services (if any) it is carrying.

2. *City Bus Traces (CBT) [66]:* Bus traces from a $2216m \times 2216m$ region exactly north of University of Washington were used. These traces capture only buses, so to better approximate the complete vehicular city traffic, we created a higher vehicular density scenario by compressing traces from different hours of the day into a single

[5]The ORBIT testbed offers improved evaluation credibility compared to standard simulation environments [118].

Table 3.2: **Default experiment parameters. For region sizes $R \geq$ 3133m, we use all available ORBIT nodes ($N$=350) and set $d$=1. This decay factor value ($f$=0.86) yields the most accurate density estimations, *i.e.,* minimizes the density mean estimation error for the default configuration. The value of $C_{thres}$ was determined empirically to minimize density mean absolute error for P3. Further increasing the size of the log does not yield significant benefits. Density estimation results for different log sizes or decay factors are not shown in the interest of space.**

| | |
|---|---|
| Region size | $2216m \times 2216m$ |
| Regional node population | N=200 nodes |
| Decay factor | f=0.86 |
| Mobility model | RWP: speeds in [5, 15] m/s |
| Spawn policy | P3, $C_{thres}$=0.6 |
| Carrier selection criterion | random |
| Estimation log size | L=4 |
| Radio range | R=250m |
| Target carrier density | d=4 |
| SA packets period | P=10sec |
| Experiment duration | 1 hour (360 periods) |
| PAS service (Section 3.5.4) spawn size | 20KB |

one-hour-long trace. Traces of buses for different hours of the day were treated as traces of different buses moving in the single hour of the experiment.

## 3.5 Evaluation

In this section, we evaluate the ability of RegReS to maintain the target density of service carriers across the design space and for different system parameters. We also evaluate the performance benefits that RegReS offers as compared to a standard epidemic scheme. Finally, we demonstrate RegReS with a small-scale deployment using PAS.

### 3.5.1 Achieving Target Density

We evaluate how well RegReS maintains the target density of carriers against several baselines:

• *Prob:* This is a standard probabilistic spawning scheme [88, 102, 157]. The value of the spawn probability ($p$=0.0016) was empirically optimized for the default configuration.

• *Static Decay Factor (SDF):* As opposed to RegReS, the decay factor of this collaborative density-estimation-based scheme is static (does not adapt) and its value ($f$=0.86) was empirically optimized for the default configuration.

• *Best Decay Factor (BDF):* The best decay factor for each configuration is determined empirically and used.

• *Oracle:* This is a hypothetical scheme. Every $P$=10 sec, the oracle, knowing exactly how many carriers are missing, makes the necessary spawns.

#### Random Waypoint Mobility

In order to provide a thorough evaluation and at the same time show the importance of adaptation, we compare the performance[4] of these schemes across different system configurations by varying one parameter at a time: 1) regional node population in Figure 3.4, 2) node speed in Figure 3.5, 3) region size in Figure 3.6, and 4) target density of carriers in Figure 3.7[6].

As shown in Figures 3.4 - 3.7, RegReS significantly outperforms Prob and SDF schemes in most scenarios. RegReS is able to adapt to varying system parameters, maintaining the targeted carrier density with less than 16% density mean absolute error and 9% mean raw error. These errors are higher compared to those for the

[6]Values of decay factor for BDF shown on top of each bar. Note that some Prob errors exceed the maximum value (50%) of the $y$-axis. Carrier density fluctuated over time for all evaluated schemes. More specifically for RegReS, the standard deviation of the carrier density is within 14% of the target carrier density.

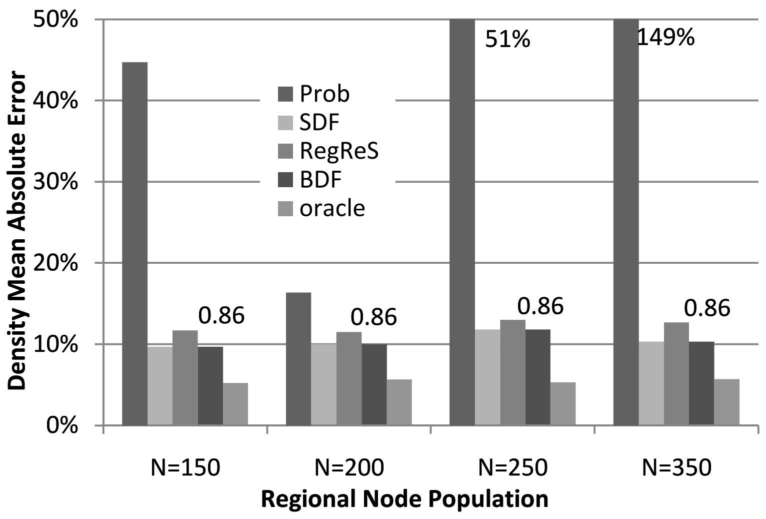


Figure 3.4: **RegReS evaluation using RWP for different node populations**[6]**.**

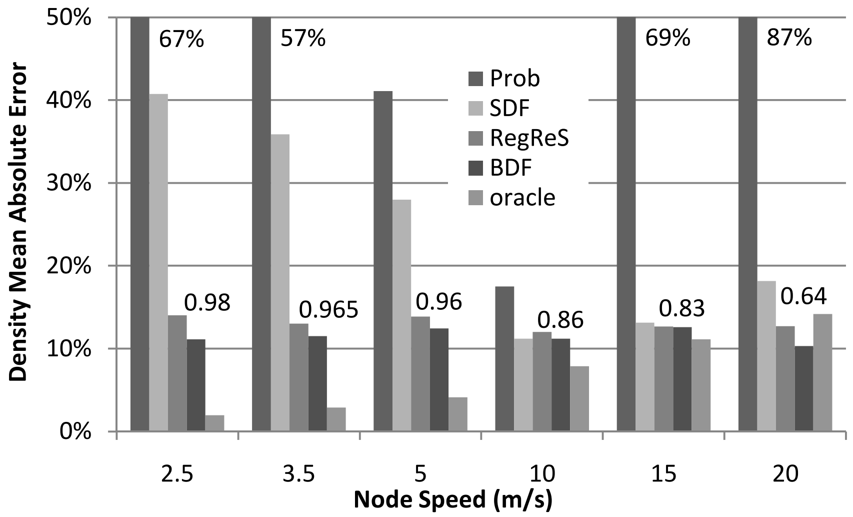


Figure 3.5: **RegReS evaluation using RWP for different node speeds**[6]**.**

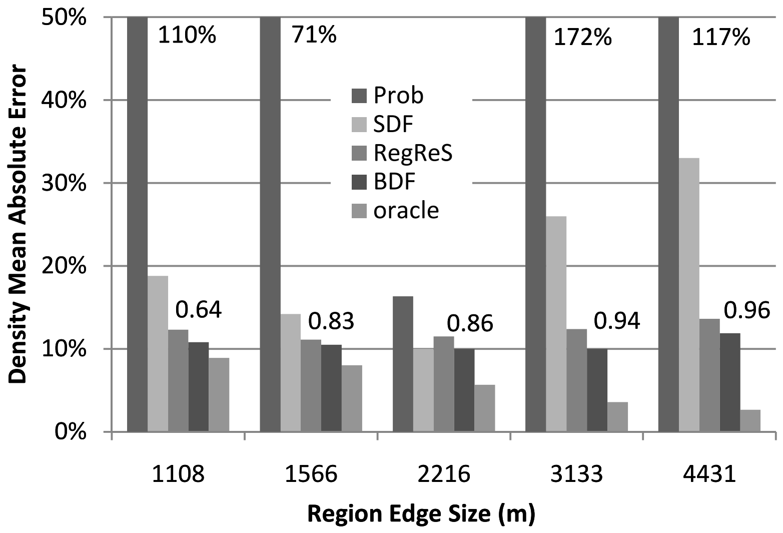


Figure 3.6: **RegReS evaluation using RWP for different region sizes[6].**

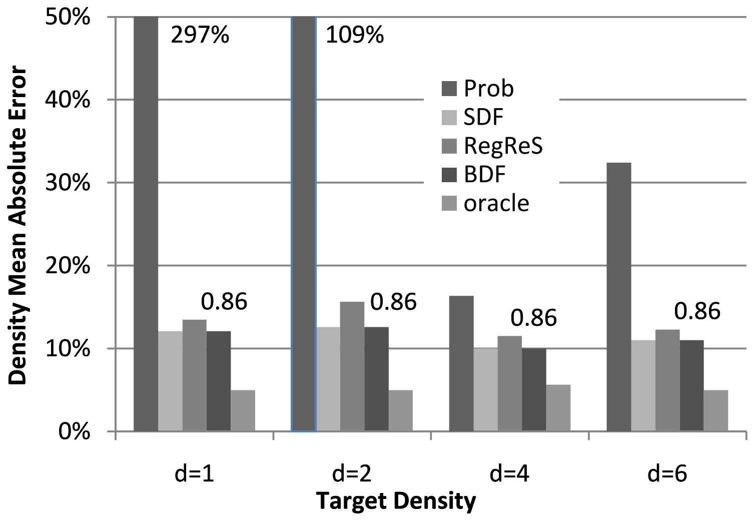


Figure 3.7: **RegReS evaluation using RWP for different target carrier densities[6].**

theoretical oracle scheme; carriers in RegReS make spawn decisions based on the limited information they collaboratively measure and exchange, without having global knowledge for the regional density. Still collaboration helps keep the errors small enough for geo-locality services that typically do not need hard guarantees. Furthermore, the performance of RegReS is very close (within 3%) to that of BDF across all configurations. For the configurations of Figures 3.4 and 3.7, SDF happens to use the best decay factor and, hence, matches the performance of BDF; the value of the best decay factor is only affected by node speed and region size as only these parameters influence the rate of carrier density changes. Prob is affected by changes in any of these four parameters.

This analysis suggests that static schemes like Prob and SDF are very weak at maintaining a target service capacity and RegReS's adaptivity is critical. An adaptive estimation scheme like that proposed by RegReS should be used in dynamic environments whether the ultimate goal is request routing, power management or service replication.

RegReS, by maintaining an accurate target density through its lightweight collaborative density estimation scheme and spawning only as many carriers as needed, can result is significant overhead savings. As Section 3.5.2 shows, the total communication overhead for RegReS, as compared to Prob and epidemic push-based schemes, is reduced by several multiples.

#### City Bus Traces Mobility

The density mean absolute errors[4], for CBT mobility and across different target carrier densities, are shown in Figure 3.8. The desired carrier density for most geo-locality services (*e.g.,* SignalGuru) is expected to be $d \geq 4$. In busy cities, such a density corresponds to a tiny fraction of the regional mobile devices. For such densities, RegReS can maintain the desired density of carriers with less than 16% mean absolute

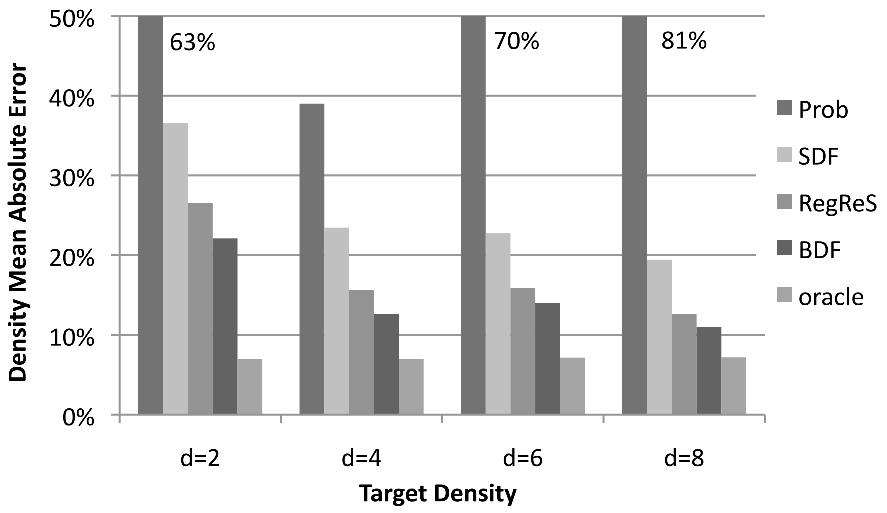


Figure 3.8: **RegReS Evaluation using CBT for different target carrier densities.**

error and 10% mean raw error (not shown in the interest of space). These errors are within 3% of those for BDF.

The schemes that are based on collaborative density estimation (SDF, BDF, RegReS) do not perform as well when very low target carrier densities are combined with CBT mobility. When node movement is highly correlated (CBT mobility), density estimation-based schemes need a high enough density of collaborating carriers to be able to build an accurate density estimate and robustly sustain clustered carrier exits. In contrast with RWP model, even a density of $d$=0.5 can be sustained.

The bus traces that we used constitute one of the hardest possible cases for RegReS. If the scenario were true instead, where all types of nodes (vehicles) were used and many more streets were traversed, nodes would be mingling better. As a result, the performance of RegReS for CBT would be closer to that for RWP.

## 3.5.2 Performance Benefits: RegReS vs. Epidemic

In this section, we evaluate the performance benefits that RegReS can offer when compared to a standard push-based epidemic scheme. As discussed, pull-based epidemic schemes or schemes based on probabilistic spawning cannot ensure that the desired density will be maintained and will often end up with too few carriers. To evaluate the performance benefits of RegReS, we use our Parking Availability Service (PAS). PAS is described in Section 3.5.4.

### Evaluation Metrics

The performance metrics that we use to compare the two schemes are:

1) **Total number of spawns:** Reflects aggregate regional resources (computation, memory, communication) spent for maintaining the target carrier density.

2) **Communication bandwidth:** Includes both spawns (20KB for PAS), service requests (66 bytes), responses (20KB for PAS) and periodic SA packets (110 bytes). Apart from communication savings, this also reflects energy savings if RegReS is running on energy-constrained devices, given that wireless network interfaces are major energy drains.

3) **Mean number of hops to reach a carrier (service access latency):** Unlike standard push-based epidemic schemes, RegReS does not proactively push the service on all nodes, and requestors have to contact a nearby service carrier. This metric reflects the delay involved for requestors to receive the service over potentially multiple hops. While the delay per hop depends on the load of the wireless channel, a single hop could involve delay below $30ms$ over 802.11a.

### Results Discussion

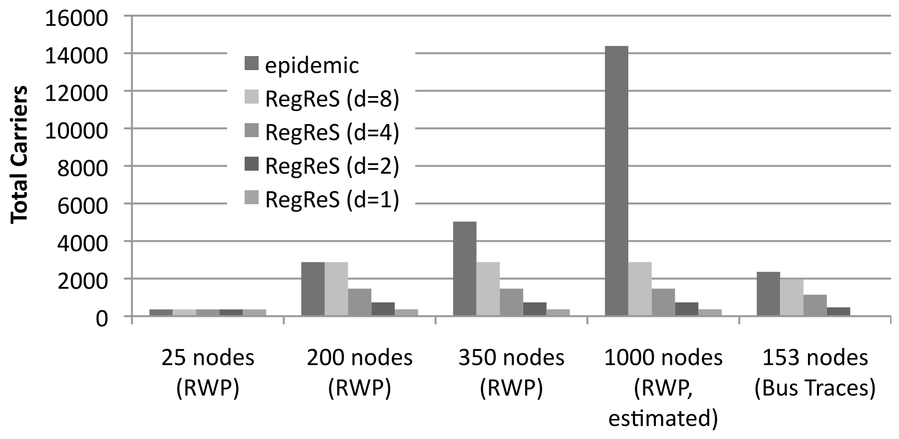
We studied target carrier densities of $d \in \{1, 2, 4, 8\}$ for different node populations of $n \in \{25, 153, 200, 350, 1000\}$ within the region. For $n$=1000 results are extrapolated based on experimental results for lower values of $n$, as ORBIT does not offer that many nodes.

**Number of spawns.** The savings that RegReS can provide in terms of total number of spawns are shown in Figure 3.9(a). The denser the region in terms of nodes and the smaller the desired carrier density the bigger the savings that RegReS provides. When $n$=25, the regional node population is very small and all schemes will have the same behavior and spawn them all; a density of $d$=1 corresponds to 25 carriers within a $2216m \times 2216m$ region.

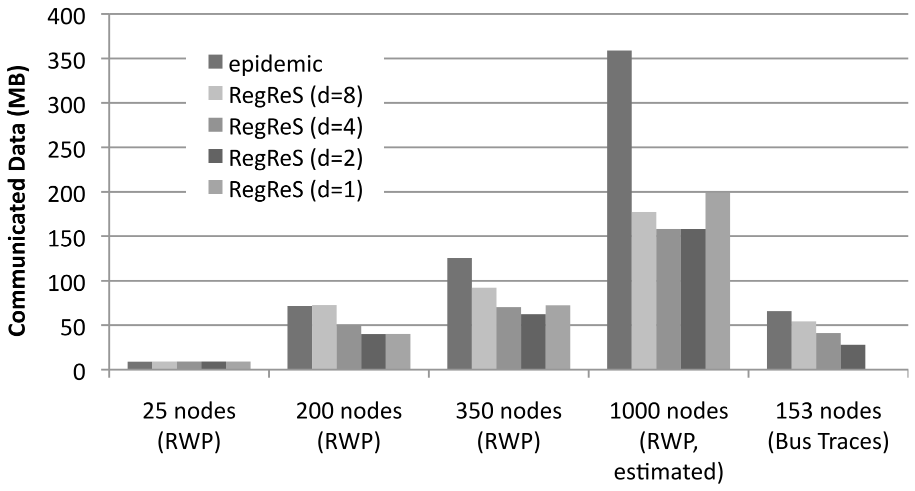
**Communication bandwidth.** RegReS, as shown in Figures 3.9(b) and (c), also helps save on total communication bandwidth. The total communication incurred depends primarily on the fraction of the nodes that get spawned or request the service (as the periodic SA packets are very small). The smaller the fraction of service requestors, the greater the savings that RegReS provides. The relative savings in the case of Figure 3.9(c) are higher than the ones in Figure 3.9(b). However, RegReS, even in the case of one third of the regional nodes requesting the service, can reduce the necessary communication by a factor of 2.3× ($n$=153).

The greater the population of nodes in the region, the greater the savings that RegReS can provide for a certain targeted carrier density as compared to the epidemic scheme. However, unlike the case of the total number of spawns metric, targeting a smaller carrier density does not always result in communication savings. Higher densities, as shown in Figure 3.10, provide higher guarantees of requestors finding the service locally and avoiding multiple hops to reach a carrier, thus reducing total communication. For most geo-locality services though, expected carrier densities, as dictated by the required sensing capacity, will be $d \gg 1$ [94], making the need for multiple hops very rare.

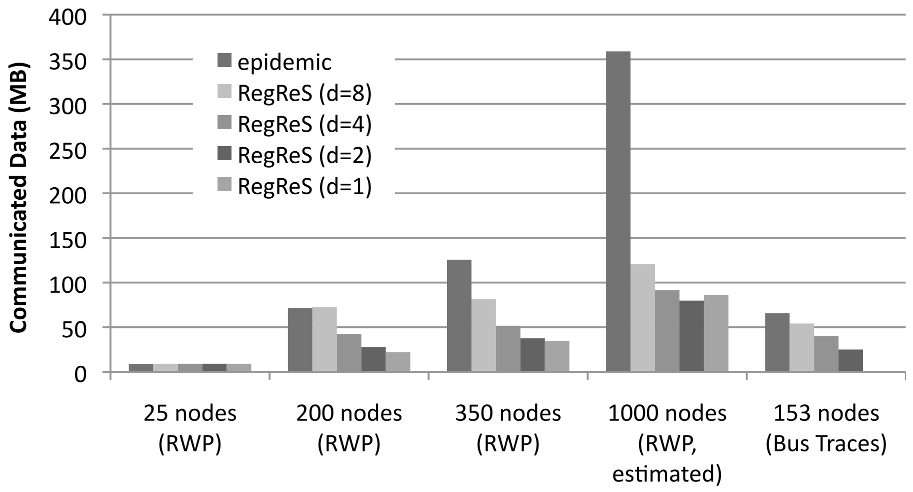
**Hops to reach a carrier (service access latency):** In Figure 3.10, the average number of hops that a requestor needs to make to receive the service is shown for the different carrier densities and mobility schemes. Due to high node mobility, the spatial distribution of carriers changes all the time, creating several transient carrier



(a)



(b)



(c)

Figure 3.9: **Total number of spawns (a), total number of bytes communicated when one third (b) and one tenth (c) of the nodes request PAS.**

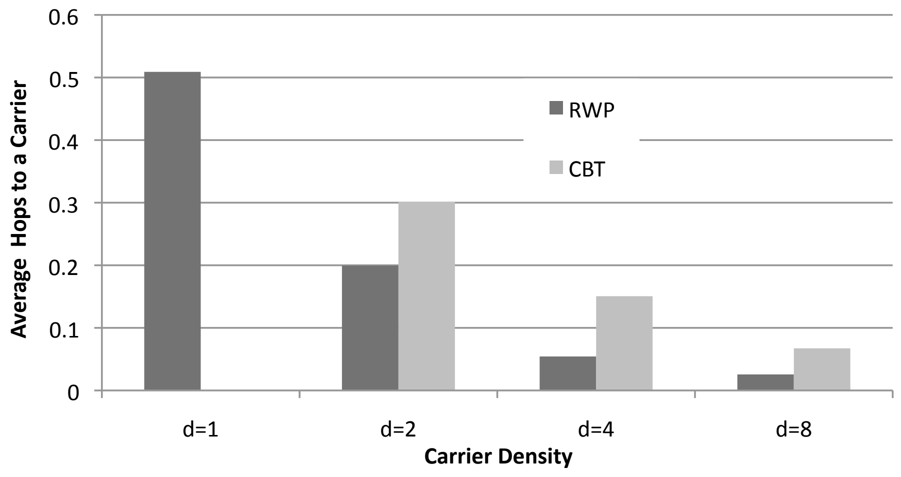
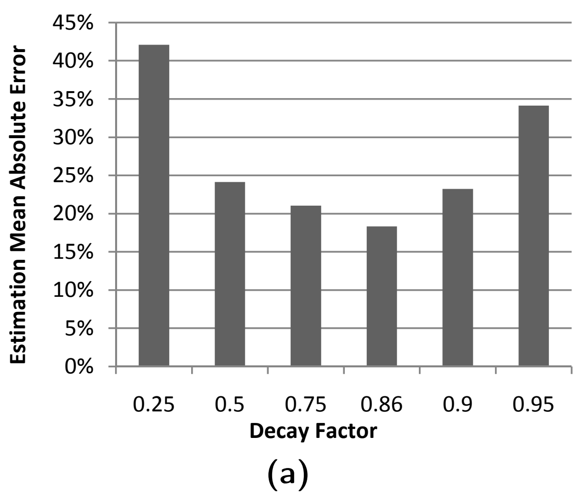


Figure 3.10: **Average number of hops needed to receive service from a RegReS carrier.**

dispersals and clusters across the region. The higher the carrier density though, the higher the probability that a requestor will receive the service from a carrier within its range and thus the smaller the average hop count. Almost never in the experimental scenarios did nodes have to traverse more than one hop. Hence, an average hop count of 0.05 (RWP, $d$=4) could be interpreted as "in ∼5% of the cases, a requestor needs to do one extra hop to receive the service and in ∼95% of them, there is a carrier in its range."

Furthermore, hop count is smaller for RWP when compared to CBT; strong movement correlation in the bus traces can lead to a lack of carriers within a node's range (carrier dispersals are more frequent), requiring hops to access a service. However, even in the case of real traces, the average hop count is quite small, allowing RegReS to provide significant savings both in terms of total number of spawns as well as communication, without incurring significant delays in the reception of a service.



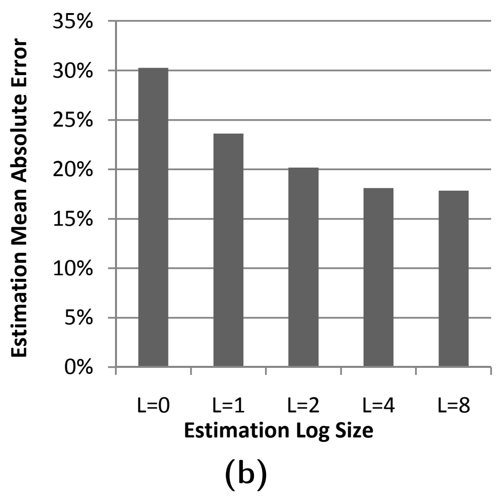


Figure 3.11: **Estimation error for different decay factors (a), log sizes(b).**

## 3.5.3 Density-Based Carrier Maintenance: Design Space Exploration

In this section, we evaluate the performance of RegReS when varying different design parameters.

#### Density estimation

Accurate estimation of the existing density of service carriers is critical in order to correctly guide spawning decisions and thus accurately maintain the target density of service carriers.

Here, we first evaluate the effect that $f$, the decay factor, has on the accuracy of the density estimation scheme. Figure 3.11(a) shows the estimation mean absolute error. This is calculated as the absolute errors of carrier merged density estimations ($DE_{merged}$) compared to the real density at each point in time, averaged across all carriers and all periods of the experiment. As shown here, the performance of the estimation scheme is greatly affected by the value of the decay factor. For our default configuration, a decay factor of $f$=0.86 gives the most accurate estimations with less than 18% mean absolute estimation error.

In Figure 3.11(b), we show the benefits of the collaborative density estimation scheme for various estimation log sizes. When no collaboration is used ($L$=0) and carriers estimate density solely based on their own measurements, the estimation mean absolute error across all carriers is 30%. With collaboration, this error almost halves, dropping to 18%. RegReS uses a log size of $L$=4, where the four most confident triplets are preserved and exchanged. Larger log sizes result in little improvement in accuracy.

### Spawn policies

In order to maintain an accurate density of service carriers, spawning decisions should be made judiciously. We evaluate how well the three spawn policies can maintain the target density. Figure 3.12 shows the density mean raw and absolute errors. The best spawn policy is P3 with a density mean absolute error of 10%. The mean raw error is even lower (8%). As discussed in Section 3.3.2, spawn policy P1 is highly spontaneous and thus ends up overspawning, leading to a density of carriers significantly higher than the target one (64% higher). Policy P2 is not as spontaneous as P1, but still not as measured in its spawn decisions as P3, as it spawns regardless of confidence. We, hence, pick P3 as RegReS's default spawn policy.

### Carrier Selection Criteria

Smarter carrier selection criteria may drastically reduce the number of spawns necessary for the maintenance of a service across its lifetime. However, carrier selection criteria that impose too strict constraints that nodes need to satisfy to become carriers, may negatively impact the accurate maintenance of the target density of service carriers. The benefits and drawbacks of the following three carrier selection criteria are evaluated:

- C1: $ERRT \geqslant NMRRT$

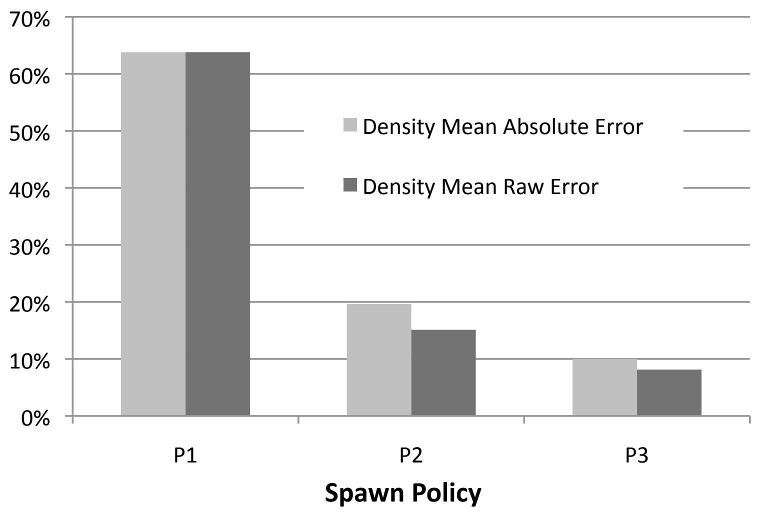
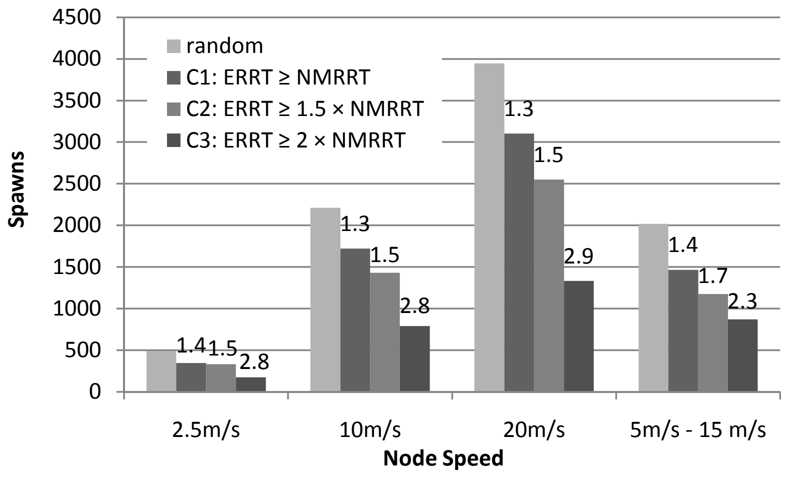


Figure 3.12: **Spawn policy evaluation.**

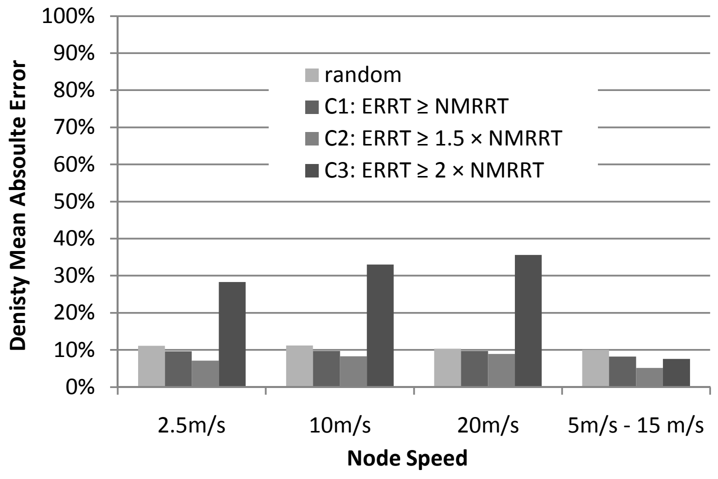
- C2: $ERRT \geqslant 1.5 \times NMRRT$
- C3: $ERRT \geqslant 2 \times NMRRT$

The number of spawns for these three criteria and the random carrier selection baseline for different node speed scenarios is shown in Figure 3.13(a). The top of each bar shows the factor by which the number of spawns is decreased compared to the random selection baseline of the same speed.

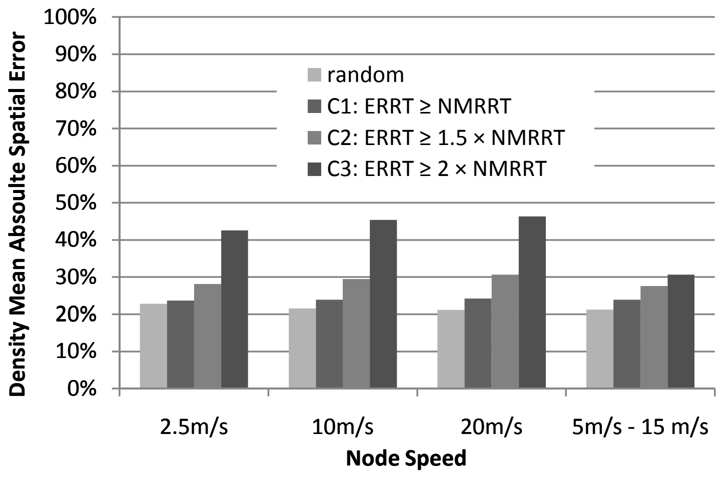
Figure 3.13(a) shows that the number of spawns can be greatly reduced by enforcing such smarter carrier selection criteria. C1, C2 and C3 reduce the number of spawns by factors of 1.3 to 1.4, 1.5 to 1.7 and 2.3 to 2.9, respectively, depending on node speed. Furthermore, Figure 3.13(b) shows that the more relaxed carrier selection criteria C1 and C2 do not hurt the carrier density maintenance as opposed to C3. C3 imposes strict constraints that nodes need to satisfy to become carriers. Therefore, existing carriers have a hard time finding eligible nodes and end up underspawning. The only exception is the case where speeds are not the same across nodes, but uniformly distributed in the range of $5m/s$ - $15m/s$. In this case, there is more diversity among nodes and, thus, carriers can easily find eligible nodes to spawn.



(a)



(b)



(c)

Figure 3.13: **Carrier selection criteria evaluation: Number of spawns (a), density mean absolute error (b), density mean absolute spatial error (c).**

The smarter carrier selection criteria C1, C2 and C3 may hurt the uniformness of the carrier distribution. To evaluate that, we use the density mean absolute spatial error metric. This is calculated by taking at each period a grid of $110 \times 110$ sampling points across the region and calculating for each gridpoint the absolute difference between the actual carrier density and the target one. These absolute differences are then averaged across all sampling points and all periods of the experiment to calculate the density mean absolute spatial error.

Figure 3.13(c) plots the density mean absolute spatial error for the four carrier selection criteria. According to Figure 3.13(c), C1 increases the density mean absolute spatial error at most by 3%, making it a very promising criterion to get savings without significant carrier distribution degradation. For C2, this error can be at most 10% and C3 may make this error even double compared to the random selection baseline.

The stricter criteria select nodes that stay longer in the region. These tend to be nodes that traverse the region mostly diagonally and/or pass close to the center. Therefore, the stricter carrier selection criteria tend to accumulate more carriers towards the center of the region and less close to the edges making the distribution of carriers less uniform.

As our analysis showed, smarter but not too strict carrier selection criteria may significantly reduce the number of spawns with only a minor degradation in the uniformness of the service carrier distribution. Spawns may be big in size, depending on the service. Carrier spawns are only 20KB for PAS, but could be up to 7.6MB for SignalGuru. Smarter carrier selection criteria can thus also significantly reduce the required communication bandwidth and energy.

### 3.5.4 Sample Application: Parking Availability Service Deployment

Previously proposed smart parking applications depend on the existence of smart parking meter infrastructure [19] or other special onboard sensors [98]. To illustrate the potential of our grassroots platform, we developed a Parking Availability Service (PAS) that does not require any additional infrastructure or hardware beyond a GPS device and vehicle-to-vehicle communications.

When a vehicle moves out of a parking spot, it broadcasts the release of the specific spot with a <Latitude, Longitude, RT> triplet. The latitude and longitude constitute the geographic location of the released spot, and RT is the Release Timestamp, *i.e.,* the time that the vehicle released the spot. The release of the parking spot is detected as a combination of the vehicle engine switching on and the vehicle gaining a speed of over $5km/h$. The former is detected with the use of a power inverter and the latter with speed information from the GPS receiver. This by no means constitutes a robust parking release detection scheme and needs further refinement.

PAS service carriers maintain a list of <Latitude, Longitude, RT> triplet entries for available spots. These entries are obtained either from vehicles while releasing parking spots or from other PAS service carriers via the epidemic service updates mechanism. An upper limit $N$ is set on the number of entries and the most recent entries (based on time elapsed since the parking spot was released) are kept.

To demonstrate PAS and the ability of RegReS to maintain such a grassroots geo-locality service, we carried out a five-node deployment using Ubuntu 8.04 laptops (four Dell Latitude D610 and one IBM Thinkpad x40) equipped with Globalsat BU-353 GPS receivers and 802.11g interfaces. The setup is shown in Figure 3.14. Three of the laptops were carried by humans and the other two were mounted on vehicles. The region was defined to be a 200 meter-long road segment and participants were asked

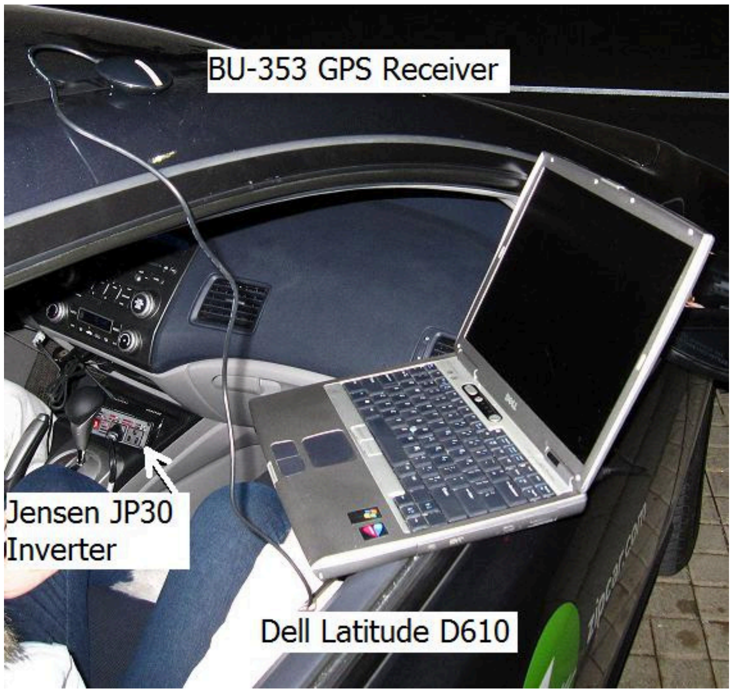


Figure 3.14: **Parking Availability Service Deployment.**

to move freely in and out. Unlike [98], our PAS is a very small-scale single-street deployment.

The service was maintained for 20 minutes using on average 2.7 carriers when the specified target density was set to three carriers within the region. To maintain this density, RegReS performed 53 spawns in total as a result of carrier exits from our small deployment area; carriers removed the service after exiting. A parking release event was also triggered after the first five minutes of the experiment. The information about the released parking spot was received by the other two carriers that were in the region at that point in time and maintained as part of the service. Two parking availability requests were also made and served in less than $30ms$.

Our PAS is a simple example that demonstrates the potential of RegReS-enabled geo-locality services. RegReS-enabled PAS service carriers maintain the information about released parking spots within the region of interest and provide the information instantaneously upon request to interested nodes over fast short-range communications.

In this way, service access latency is significantly reduced (less than $30ms$, as opposed to several seconds over 3G) and the use of the costly, scarce and potentially unavailable long-range communications is avoided.

## 3.6 Conclusions

To completely alleviate the dependence of geo-locality services on cloud infrastructure and long-range communications, this chapter presented the RegReS middleware. The RegReS middleware enables the full hosting of geo-locality services on confederations of mobile devices that collaborate and serve requests over fast short-range ad hoc communications. In this way, RegReS obviates the need for costly and slow long-range communications. Unlike previously proposed schemes for geo-locality service maintenance, RegReS allows services to specify not only their region and lifetime but also their desired carrier density. In this way, RegReS engages only as many nodes as specified in the provision of a service and avoids wasting the increased, yet constrained, resources of mobile devices.

To ensure accurate service carrier density maintenance in highly dynamic and unreliable environments, RegReS employs a fully-distributed collaborative scheme. Service carriers opportunistically collaborate to estimate the current service density and spawn additional carriers, where necessary, directed by a confidence-based policy. Thanks to collaboration, RegReS carriers make twice as accurate density estimations. Collaboration is thus very important in highly dynamic and unreliable environments.

At the same time, RegReS adapts to different system parameters by using the CMRT metric as a proxy for system dynamics. By adapting to system dynamics, RegReS manages to effectively maintain the required service density across a wide range of environments with less than 10% mean raw error and 16% mean absolute error. As our results show, without adaptation, both errors could be arbitrarily high, up to 41%. Thus, adaptation too is critical in highly volatile mobile environments.

Being able to maintain an accurate target density of service carriers in a fully-distributed and lightweight approach, RegReS constitutes an efficient foundation for low- or even zero-infrastructure geo-locality services in mobile ad hoc networks. Our PAS constitutes a simple example that demonstrates the benefits of RegReS-enabled geo-locality services. As an example of a more challenging service that can demonstrate the high capabilities of collaborative computing platforms like RegReS, the next section describes SignalGuru, a collaborative traffic signal schedule advisory service.

# Chapter 4

# Camera-based Geo-locality Services: SignalGuru

The previous chapter presented RegReS, a middleware that enables the hosting of geo-locality services on confederations of collaborating mobile devices. In this chapter, to demonstrate the potential of collaborative mobile device-based computing platforms enabled by middleware like RegReS, we describe several novel services that they can support. More specifically, we focus on novel smartphone camera-based services that are most challenging because of the increased computational resources (*e.g.,* CPU for image processing) that they necessitate. This chapter explores the challenges faced when building smartphone camera-based geo-locality services and showcases SignalGuru, a novel traffic signal schedule advisory service. The example of SignalGuru demonstrates that with collaboration, adaptation and by leveraging other capabilities of smartphones (*e.g.,* accelerometer and gyro), even challenging camera-based services can be fully supported on confederations of mobile devices without the need for cloud servers and costly long-range communications to reach them.

## 4.1 Introduction

With an ever richer set of sensors, higher computational power and greater popularity, smartphones have become a major collaborative sensing platform. In particular, smartphones have been widely used to sense their environment and provide services to assist drivers. Several systems have been proposed that leverage smartphones' GPS, accelerometer and gyro sensors in order to estimate traffic conditions [55, 99, 145], detect road abnormalities [99] and compute fuel-efficient routes [44].

Cameras, in contrast to other smartphone sensors, have so far been underutilized for automated collaborative sensing. Cameras have been used only for a handful participatory sensing systems; both image capture and image analysis are performed by a human user. Such applications include the monitoring of vegetation, garbage, and campus assets [120]. In all these services, the user needs to point his smartphone camera to the target object, capture an image and upload it to the central service where a human operator analyzes it. The proposal of collaborative sensing services that leverage smartphone cameras without manual user effort has so far been hindered by two false beliefs: 1) the view of smartphone cameras is always obstructed (carried in pockets or placed flat on the table) and 2) image processing requirements are prohibitively high for resource-constrained mobile devices.

In this chapter, we propose a novel collaborative sensing platform that is based on the cameras of windshield-mounted smartphones. We show that near real-time and accurate camera-based services are possible on confederations of such mobile devices. Several drivers are already placing their phones on the windshield in order to use existing popular services like navigation. Once a phone is placed on the windshield, its camera faces the road ahead. Our proposed sensing platform leverages these cameras to opportunistically capture content-rich images of the road and the environment ahead. Inter-device collaboration is also leveraged to gather more visual road-resident information and distill it into knowledge (services) that can be provided to the drivers.

With their cameras, a network of collaborating windshield mounted smartphones can enable a rich set of novel services.

More specifically, in this chapter, we focus on the description and evaluation of the SignalGuru service [85]. SignalGuru leverages the cameras of windshield-mounted smartphones in order to detect traffic signals ahead and predict their future schedule. SignalGuru devices collaborate with other regional devices in order to mutually improve their historic traffic signal status information and better predict when the signal ahead will turn green/red.

Besides the challenges for RegReS-enabled geo-locality services mentioned in Section 3.2.1, providing real-time camera-based services, like SignalGuru, on top of a confederation of collaborating windshield-mounted smartphones poses several additional challenges that need to be tackled:

1. *Commodity cameras*: The quality of smartphones' cameras is significantly lower than that of high-end specialized cameras used in computer vision and autonomous navigation. Smartphone cameras have both lower color quality and lower resolution. Further, as the capturing of still images is very slow (1-2 seconds) on an iPhone 3GS device, video frames should often be used instead for low-overhead and high-frequency image-based detection. This further degrades resolution, as video resolution is only up to 640×480 pixels for iPhone 3GS and 1280×720 pixels for iPhone 4 devices.
2. *Limited processing power*: Processing video frames to detect visual information takes significant computational resources. In SignalGuru, traffic signal detection is the most compute-intensive task. A traffic signal detection algorithm that runs on resource-constrained smartphones must be lightweight so that video frames can still be processed at high frequencies. The higher the processing frequency, the more accurately SignalGuru can measure the duration of traffic signal phases and the time of their status transitions.

3. *Uncontrolled environment composition and false detections*: Windshield-mounted smartphones capture the real world while moving. As a result, there is no control over the composition of the content captured by their video cameras. Results from one of our deployments suggest that the camera-based traffic signal detection algorithm can confuse various objects for traffic signals and falsely detect traffic signal colors. A misdetection rate of 4.5% can corrupt up to 100% of traffic signal predictions. Schemes need to be devised to carefully filter noisy image-based object detections.
4. *Variable ambient light conditions*: Still image and video capture are significantly affected by the amount of ambient light that depends on both the time of the day and the prevailing weather conditions. By adjusting the camera exposure time to the fixed luminous intensity of traffic signals, SignalGuru robustly detects traffic signals regardless of the prevailing ambient light conditions.
5. *Need for collaboration*: The visual information that an individual smartphone senses is limited to its camera's view angle. Regional smartphones thus need to collaborate in order to increase their information reach. In the SignalGuru service, for example, a device may not be able to see a far-away traffic signal, or may not be within view of the traffic signal for a long enough stretch of time. Collaboration is needed between vehicles in the vicinity (even those on intersecting roads) so that devices have enough information to be able to predict the schedule of traffic signals. Collaboration is also needed in order to maintain SignalGuru's data over time and in a distributed fashion within the vehicular network. This last challenge is tackled by collaborative middleware like RegReS.

It should be noted that we do not consider battery lifetime as a major challenge. Mobile phones can be plugged into the ample energy resources of a vehicle. In cases where this does not hold, approaches proposed for lifetime maximization in sensor networks [75, 156] can be used. Such approaches can determine if and when a given device needs to perform certain power-hungry tasks (*e.g.,* SignalGuru traffic signal detection, collaboration with wireless communication).

The research in this chapter makes three major general contributions and a more specific one:

1. *RegReS services:* The example of SignalGuru illustrates the potential of smartphone-based collaborative computing platforms that are enabled by middleware like RegReS. It demonstrates that such platforms can fully support a rich set of services, including challenging and computationally intensive camera-based services, *without* the need for cloud servers and costly long-range communications to them. Besides SignalGuru, the potential of four more novel services is discussed.
2. *Collaboration and adaptation:* The case of SignalGuru highlights the importance of both collaboration and adaptation for grassroots RegReS services. With collaboration and adaptive Support Vector Regression (SVR) models, not only pre-timed but also state-of-the-art traffic-adaptive traffic signals can be predicted with very small mean absolute error ($2.45s$). Collaboration reduces the error by 78% and adaptation by an additional 25%.
3. *Leveraging mobile device capabilities:* We show that networks of windshield-mounted smartphones can greatly increase their camera-based sensing frequency and accuracy by fusing information from the smartphone's Inertial Measurement Unit (IMU) to reduce the video area that needs processing. Our IMU-based detection window halves both the processing time and the misdetection rate for SignalGuru. We also propose and evaluate low-pass filtering and a colocation filter that effectively filter away false positive event (*e.g.,* traffic signal transition) detections. Thanks to these proposed approaches, camera-based services like SignalGuru become both computationally tractable and accurate on confederations of commodity smartphones.

4. *SignalGuru benefits:* We propose five user-focused applications that can be built on top of SignalGuru. These applications can help drivers reduce their fuel consumption, environmental impact and travel time. In particular, our SignalGuru-enabled Green Light Optimal Speed Advisory (GLOSA) application offers speed advisories to avoid undue stops and waits at red lights. Testing this system using an onboard fuel efficiency monitor, we show that when drivers follow the advisory of our GLOSA system, 20.3% fuel savings can be achieved.

In the next sections, we describe SignalGuru in detail. In Section 4.2, we present the motivation behind SignalGuru and in Section 4.3, the SignalGuru-enabled GLOSA application. Section 4.4 describes the operation of traffic signals and Section 4.5 the architecture of our collaborative SignalGuru service. In Section 4.6, we present our experimental methodology and in Section 4.7, we evaluate the performance of SignalGuru's individual modules based on our two real-world deployments. Section 4.8 discusses the operation of SignalGuru in complex intersections. In Section 4.9, we discuss four more applications, in addition to GLOSA, that are enabled by SignalGuru's traffic signal schedule predictions. Besides SignalGuru and its enabled applications, Section 4.10 describes four more services that windshield-mounted smartphones could support and discusses their challenges. Lastly, Section 4.11 surveys related work.

## 4.2 SignalGuru Motivation

With more than 272,000 traffic signals in major intersections of the USA alone [79], our daily driving experience is significantly influenced by them. Traffic signals are widespread in developed countries as they allow competing flows of traffic to safely cross busy intersections. Traffic signals, however, do take their toll. The stop-and-go movement pattern that they impose increases fuel consumption by 17% [5], $CO_2$

emissions by 15% [5], causes congestion [21], and leads to increased driver frustration [79].

Drivers can be assisted with a GLOSA system [5, 149]. A GLOSA system advises drivers on the optimal speed they should maintain when heading towards a signalized intersection. Should drivers maintain this speed, then the traffic signal will be green when they reach the intersection, allowing the driver to cruise through.

Worldwide, only a handful of GLOSA systems have been deployed [149], and have so far been based on roadside message signs (wired to traffic signals). These signs are placed a few hundred meters away from the signal and display the optimal speed drivers should maintain. Their costly and often impractical deployment and maintenance, however, has hindered their widespread usage.

Countdown timers at vehicular traffic signals constitute another alternative approach to assist drivers; digital timers next to the traffic signal display the time till the signal changes from red to green and vice versa. Such traffic signals are deployed only in a few cities around the world. The cost of updating existing traffic signals to include such timers has hindered their widespread deployment.

Countdown timers for pedestrian traffic signals are much more common in the USA and the rest of the world, and drivers can sometimes use these to infer when the light will turn green. However, very often these are not visible from far away but only after one has reached the intersection. At that time, it is too late for drivers to adapt their speed and so they need anyway to come to a complete halt. Furthermore, at some intersections, it is not easy or even possible for the driver to infer the time the signal will switch; the intersection may have a complex phase schedule and the green light for the driver may not come straight after some pedestrian timer counts down to zero.

US and European transportation agencies recognize the importance of GLOSA and access to traffic signal schedules, and thus have advocated for the integration of short-range (DSRC) antennas into traffic signals as part of their long-term vision [21, 123]. DSRC-enabled traffic signals will be able to broadcast in a timely fashion their schedule to DSRC-enabled vehicles that are in range. Audi recently prototyped a small-scale DSRC-based GLOSA system for 25 traffic signals in Ingolstadt (Germany) [5]. The widespread deployment of such an approach, however, has been hindered by the significant cost to equip traffic signals and vehicles with the necessary specialized computational and wireless communications infrastructure.

In this work, we take an *infrastructure-less* approach, demonstrating that the proposed collaborative platform of windshield-mounted smartphones and their cameras can be leveraged to access traffic signal schedules. Windshield-mounted smartphones use their cameras to detect and determine the current status of traffic signals. Multiple phones in the vicinity use opportunistic ad-hoc communications to collaboratively learn the timing patterns of traffic signals and predict their schedule. SignalGuru's predicted traffic signal schedule then enables GLOSA and other possible applications on the phone.

## 4.3 SignalGuru Applications: GLOSA

The goal of the GLOSA application is to advise drivers on the optimal speed they should maintain so that the signal is green when they arrive at the next intersection. In this way, the driver can cruise through the intersection without stopping. A GLOSA application can offer several benefits, such as 1) decreased fuel consumption [5], 2) smoothed and increased traffic flow (stop-and-go patterns avoided) [21], and as a result of these, 3) decreased environmental impact [5].

A GLOSA application needs four pieces of information in order to calculate the optimal speed: 1) the residual amount of time till the traffic signal ahead turns green, 2) the intersection's (stop line) location, 3) the vehicle's current location, and 4) the queue length of the traffic signal ahead. The first is provided by SignalGuru, the second by map information [110] and the third by the available localization mechanisms on the mobile device (*e.g.,* GPS). The traffic signal queue length can be estimated by fusing information about the number and positions of vehicles in the queue as described in [30]. Then the time it takes for the queue ahead to discharge can be calculated as a function of the queue length [79].

If no traffic signal queue length information is available, and when vehicles are very close ($<100m$) to the intersection, GLOSA should switch from a speed advisory to a time countdown (till the signal ahead turns green). Drivers can then look at the queue length ahead and manually estimate their optimal speed.

Although GLOSA may often advise a vehicle to reduce its speed, the vehicle's total travel time will not be increased. On the contrary, it may get decreased. Despite the speed reduction, a GLOSA-enabled vehicle will still travel through the intersection at the same traffic signal phase as it would if it were traveling at its regular speed. Moreover, at the time the signal turns green, a GLOSA-enabled vehicle will be cruising through the intersection with an initial non-zero speed, as opposed to a regular vehicle that would have to start from a complete halt. Therefore, GLOSA may improve an individual vehicle's travel time.

GLOSA also improves the overall flow reducing congestion. The traffic flow is smoother and faster when vehicles are cruising through the intersections as opposed to when they come to a complete halt and then slowly accelerate one after the other to cross the intersection. Traffic flow improvements then lead to further gas and travel time savings.

The larger the available lead-up time, *i.e.,* the amount of time in advance at which predictions are available, the more effective GLOSA is. Predictions that are available say 20 seconds in advance, while the driver is perhaps $250m$ from the traffic light, provide enough room to control the vehicles' speed. The prediction accuracy should be less than  10% of a traffic signal phase length to avoid wasting precious green time (*e.g.,* not guiding a vehicle to the intersection long after the light has switched to green).

Besides GLOSA, SignalGuru's traffic signal schedule predictions can be used to enable more applications to help drivers reduce fuel consumption, reduce environmental impact, reduce travel time and increase safety. Four such additional applications are described in Section 4.9.

## 4.4 Traffic Signal Background

In signalized intersections, different but non-conflicting (safe to co-exist) vehicular and pedestrian movements are grouped together to run at the same time. Such groups of movements are termed phases. Figure 4.1 shows a typical simple intersection with two phases. When the light is green for phase A, vehicles or pedestrians moving North-South can safely move at the same time. Later the traffic signal will turn red for phase A and green for phase B. At this time, vehicles and pedestrians moving East-West can go. When this phase completes, the intersection has completed one *cycle* and the light will turn red again for phase B and green for phase A. Many intersections may have more than two phases. The amount of time that the light stays green in a given phase is *phase length.* The sum of all phase lengths of an intersection is *cycle length.*

Most traffic signals in the US are pre-timed traffic signals [122]. For pre-timed traffic signals, the settings (phase lengths, cycle length) of the traffic signals are fixed and the exact same schedule repeats in every cycle. The settings only change when the intersection switches mode of operation depending on the day or the time of day. Typically pre-timed traffic signals have three modes of operation: 1) off-peak, 2) a.m. peak and 3) p.m. peak. Sometimes, there is a special schedule for Saturday peak.

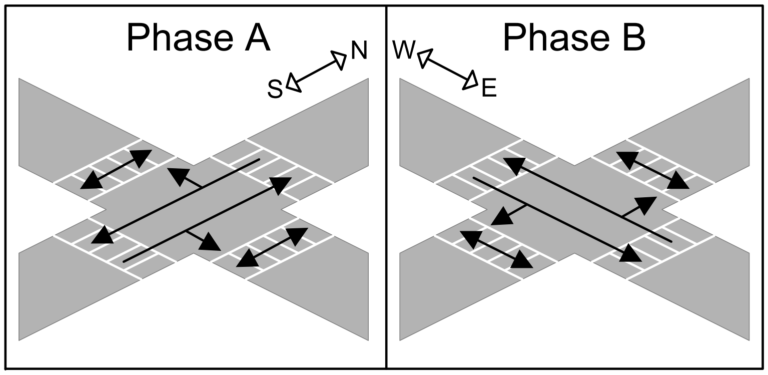


Figure 4.1: **The two phases of a typical right-hand traffic intersection.**

In contrast to the US, Singapore uses the state-of-the-art GLIDE system that is adapted from the SCATS [132] system to adaptively control its traffic signals. SCATS controls traffic signals in 144 cities around the world and adaptively adjusts settings based on measurements from its inductive loop detectors. One loop detector is installed per lane and placed beneath the road surface at the intersection stop line. Loop detectors, while the light is green, measure the saturation of their lane. Specifically, lane saturation is calculated as a function of the number of vehicles that traveled over the corresponding loop detector and the measured total gap time (*i.e.,* amount of time that the loop detector is unoccupied). Lane saturations are merged to calculate a phase's saturation.

SCATS adjusts traffic signal settings in order to balance the saturation across the different phases of the intersection. The higher the saturation of a phase (more vehicles), the greater portion of the cycle length is allocated to the specific phase. Cycle length duration is adjusted depending on the saturation of all the phases of the intersection and increases when the maximum phase saturation increases. Longer cycles allow intersections to operate more efficiently (higher throughput), but increase the waiting times and frustration of drivers. SCATS measures phase saturations and changes the intersection traffic signal settings accordingly every cycle, *i.e.,* every 1-3 minutes.

## 4.5 SignalGuru Architecture

SignalGuru aims to detect and predict the schedule of traffic signals using just software on commodity smartphones. It is a grassroots software service that leverages opportunistic sensing on mobile phones to detect the current color of traffic signals, share with nearby mobile phones to collectively derive traffic signal history, and predict the future status and timing of traffic signals.

Figure 4.2 shows the modules in the SignalGuru service. First, phone cameras are used to capture video frames, and detect the color of the traffic signal (detection module). Then, information from multiple frames is used to filter away erroneous traffic signal transitions (transition filtering module). Third, nodes running the SignalGuru service broadcast and merge their traffic signal transitions with others in communications range (collaboration module). Finally, the merged transitions database is used to predict the future schedule of the traffic signals ahead (prediction module).

The prediction of the future schedule of traffic signals is based on information about past timestamped R$\rightarrow$G transitions, *i.e.,* information about when the traffic signals transitioned from red to green in the current or previous cycles. The prediction is based on R$\rightarrow$G transitions, as opposed to G$\rightarrow$Y (green to yellow) transitions, because vehicle-mounted smartphones can witness and detect R$\rightarrow$G transitions much more frequently; when the traffic signal is red, vehicles have to stop and wait till the signal turns green. As a result, it is quite likely that a vehicle will be at the intersection at the moment that the R$\rightarrow$G transition occurs and thus detect it. For a G$\rightarrow$Y transition to be detected, the vehicle needs to be driving towards the intersection and have good view ($\leq$ 50 meters away) of the signal when the signal color changes. As a result, it is much less likely[1] for a vehicle to be close enough to the intersection at the moment

[1] In our Singapore deployment, vehicles witnessed in total 37 R$\rightarrow$G transitions but only two G$\rightarrow$Y transitions.

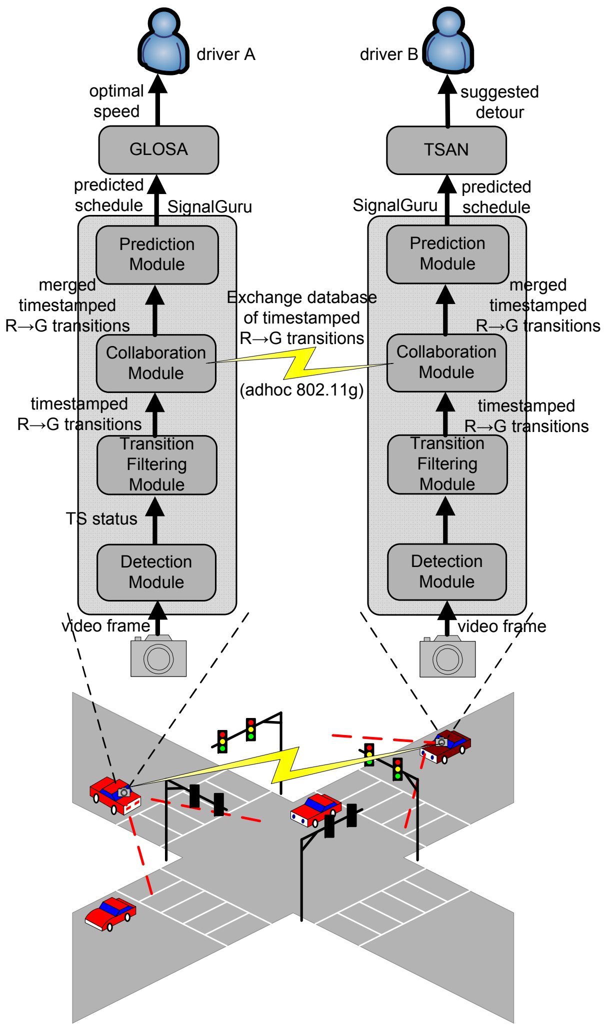


Figure 4.2: **SignalGuru service architecture. The two SignalGuru-enabled vehicles are collaborating to predict the schedule of the traffic signals. One vehicle feeds SignalGuru's schedule predictions into GLOSA and the other into the Traffic Signal Adaptive Navigation (TSAN) application. TSAN is described in Section 4.9.**

of the G→Y transition. The same applies also for Y→R transitions. Section 4.5.4 discusses how timestamped R→G transition information is used to predict the traffic signal schedule.

In the next sections, we describe in detail the design of the various modules of SignalGuru's software architecture and the way SignalGuru tackles the challenges mentioned in Section 4.1.

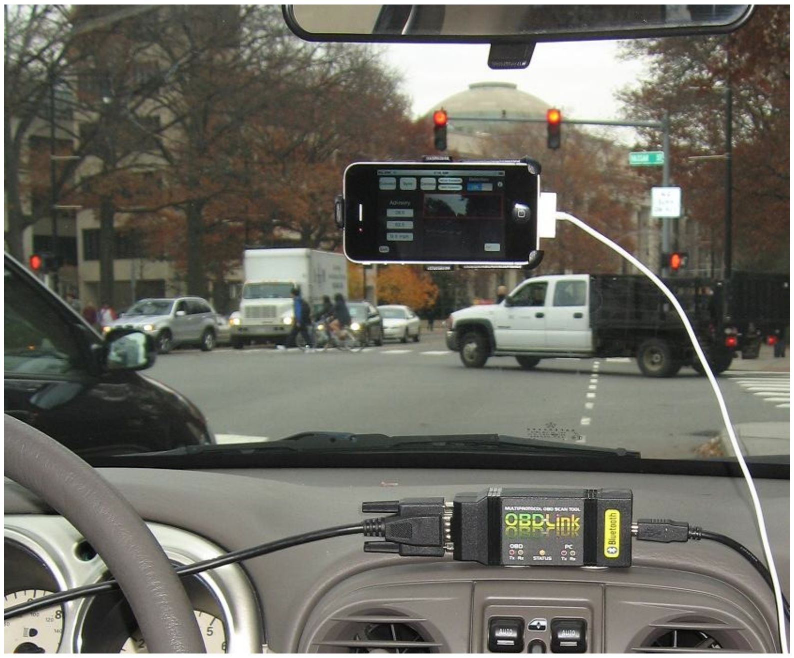


Figure 4.3: **SignalGuru-enabled iPhone mounted on the windshield. The OBD-LINK device used to measure fuel consumption is also shown.**

### 4.5.1 Detection Module

The detection module detects and reports the current color of potential traffic signals in the captured video frames. The detection module is activated based on its GPS location[2] and only when it is close ($<50m$) to a signalized intersection. The video frames are captured using the standard iPhone camera. As Figure 4.3 shows, when the smartphone is mounted on the windshield, this camera is facing outside and thus able to capture videos of the traffic signals ahead. This is just as users would mount their smartphone when using a navigation or other travel-related application.

#### Detection Algorithm

SignalGuru's traffic signal detection module must be lightweight and fast so that the color of traffic signals can be sensed as frequently as possible, and the time of transitions is detected as precisely as possible. The time accuracy of color transition

[2]We configure the iPhone's GPS to return location stamps of the maximum possible accuracy and frequency.

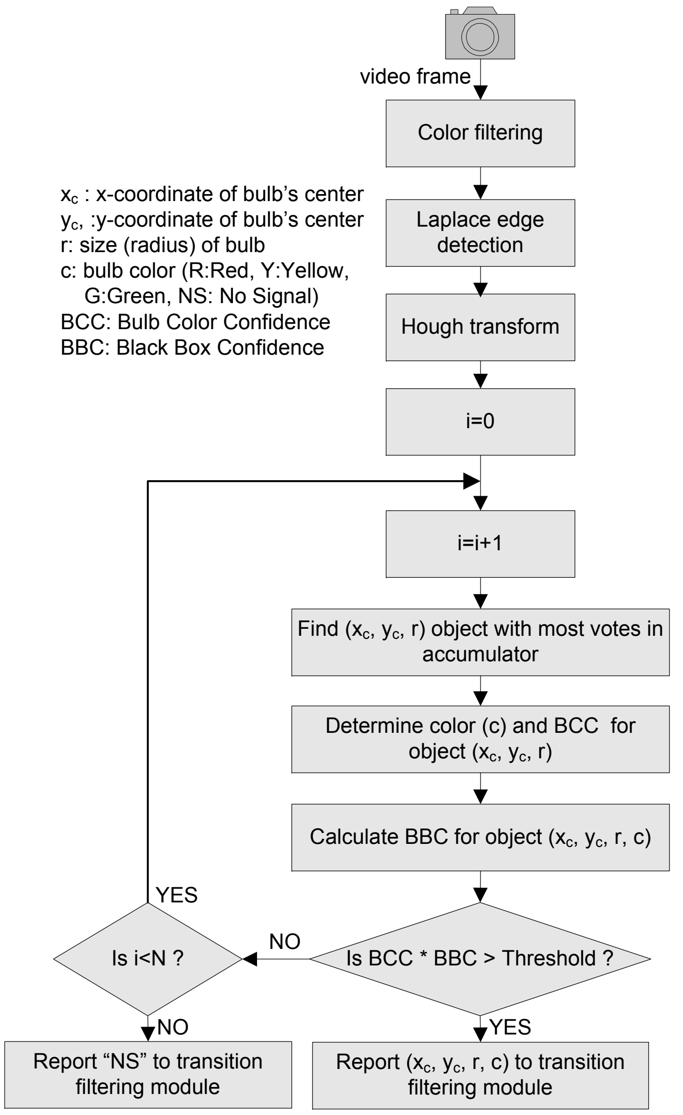

video frame
Color filtering
Laplace edge detection
Hough transform
i=0
i=i+1
Find (xc, yc, r) object with most votes in accumulator
Determine color (c) and BCC for object (xc, yc, r)
Calculate BBC for object (xc, yc, r, c)
Is BCC * BBC > Threshold ?
NO
YES
Is i<N ?
YES
NO
Report "NS" to transition filtering module
Report (xc, yc, r, c) to transition filtering module
xc : x-coordinate of bulb's center
yc, :y-coordinate of bulb's center
r: size (radius) of bulb
c: bulb color (R:Red, Y:Yellow, G:Green, NS: No Signal)
BCC: Bulb Color Confidence
BBC: Black Box Confidence

Figure 4.4: **Traffic signal detection algorithm. "NS" stands for "No Signal" and is the status returned by the detection module when no traffic signal can be detected with a confidence higher than the threshold value.**

detections directly affects the time accuracy of predictions, as Section 4.5.4 explains. Our SignalGuru detection module is able to process a fresh frame every two seconds.

Figure 4.4 shows our image processing algorithm used to process video frames for the purpose of detecting traffic signals. The algorithm is based on the three most characteristic features of a traffic signal, which are the bright red/yellow/green color of its bulbs, the shape of its bulbs (*e.g.,* circle, arrow) and its surrounding black box (traffic signal housing).

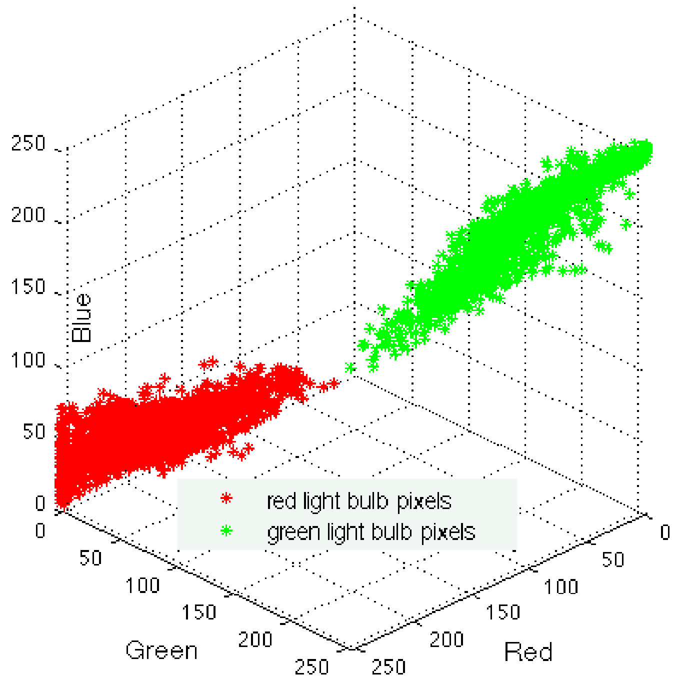


Figure 4.5: **Traffic signal bulb color distribution in the RGB color space. 2628 and 2326 pixels are drawn from 200 red and 200 green traffic signals in Cambridge, USA, respectively.**

The first step of the detection algorithm is the color filtering process, as the most distinctive feature of traffic signals is the bright color of their bulbs. The color filter inspects the color of all pixels of an image (video frame) and zeroes out the pixels that could not belong to a red, yellow or green traffic signal bulb. Thus, the color-filtered image contains only objects that have the correct color to be a traffic signal bulb. The color filter was designed empirically by analyzing the color range of red, yellow, green bulb pixels from a set of 200 traffic signal pictures for each color[3]. The color distribution for red and green traffic signal bulbs is shown in Figure 4.5. The color filter is relatively lightweight computationally when performed in the device native color space (*i.e.,* RGB), and also manages to zero out most of an image, reducing computing needs in subsequent stages. For all these reasons, the color filtering stage comes first.

[3]We used a different color filter for Cambridge and Singapore as the two cities use traffic signals implemented with different technologies.

After color filtering, only objects that have the correct color are maintained in the image. The next few stages examine which of them qualify to be a traffic signal based on their shape (*e.g.,* circle, arrow). This is achieved by first applying a Laplace edge detection filter that highlights the boundaries of the color filtered objects and then a Hough transform. The Hough transform uses a voting mechanism (accumulator) to decide which objects constitute the best traffic signal bulb candidates based on their shape.

Once the Hough transform voting is completed, the accumulator determines which object has the most votes and is thus the best candidate to be a traffic signal. The accumulator contains information about the location of the best candidate in the image as well as its size (*e.g.,* radius).

Then, the pixels of the candidate area are inspected to decide on the color of the bulb and count exactly what percentage of the pixels falls into the correct color range. This percentage is termed the *Bulb Color Confidence (BCC)*. BCC helps to avoid confusing, for example, road signs with a circular red perimeter but a different color in the center (*e.g.,* right turn prohibited sign) as a red signal.

According to the color and size of the bulb, a specific area around the bulb is checked for the existence of a horizontal or vertical black box, the traffic signal housing. For example if the bulb is red, the area below or on the left is searched for a vertical or horizontal traffic signal black box, respectively. A *Black Box Confidence (BBC)* metric is also used based on how many pixels of the searched area are dark enough to qualify as traffic signal black box pixels.

The product $BCC \times BBC$ constitutes the *detection confidence* for a specific object in the video frame. If the detection confidence is higher than a threshold value, then the detection is considered valid and the traffic signal with the detected color is reported. If not, then the next best candidate from the Hough transform accumulator is examined. We found that a detection confidence threshold of 0.6 yielded the lowest

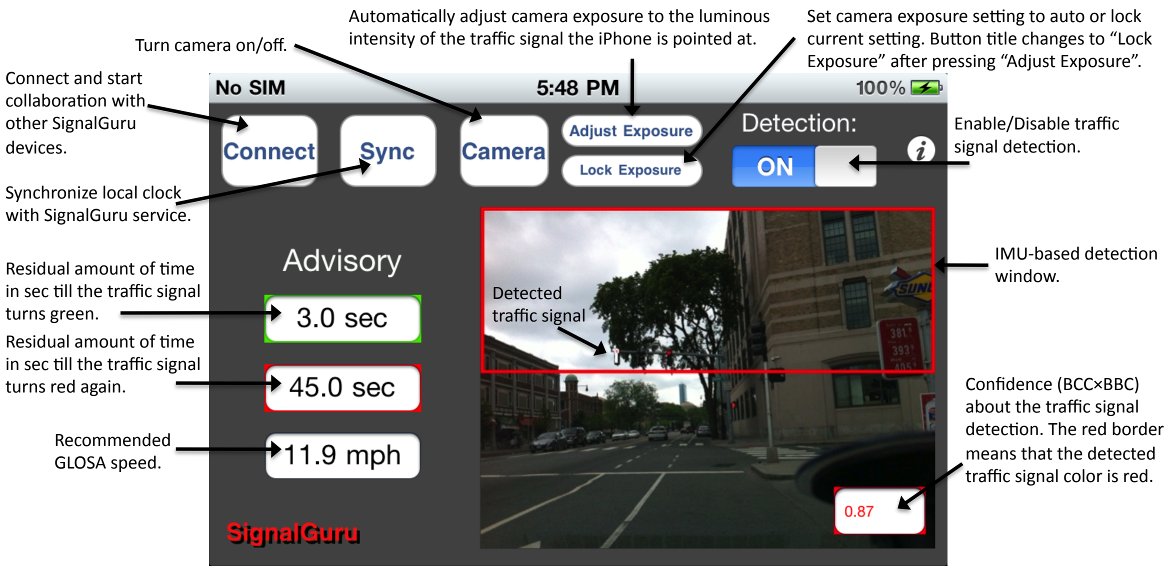

Automatically adjust camera exposure to the luminous intensity of the traffic signal the iPhone is pointed at.
Set camera exposure setting to auto or lock current setting. Button title changes to "Lock Exposure" after pressing "Adjust Exposure".
Turn camera on/off.
Connect and start collaboration with other SignalGuru devices.
Synchronize local clock with SignalGuru service.
No SIM
5:48 PM
100%
Connect
Sync
Camera
Adjust Exposure
Lock Exposure
Detection:
ON
Enable/Disable traffic signal detection.
Advisory
Residual amount of time in sec till the traffic signal turns green.
3.0 sec
Residual amount of time in sec till the traffic signal turns red again.
45.0 sec
Recommended GLOSA speed.
11.9 mph
SignalGuru
Detected traffic signal
IMU-based detection window.
Confidence (BCC×BBC) about the traffic signal detection. The red border means that the detected traffic signal color is red.
0.87

Figure 4.6: **SignalGuru service screenshot. GLOSA advisory has also been included in the same iPhone application. Audio advisory can complement the visual advisory to alleviate driver distraction.**

detection false positive and false negative rates for our database (400 pictures). We also found that there is little additional value in inspecting more than the 10 best candidates of the Hough voting mechanism. As a result, the looping criterion in Figure 4.4, $N$, is set to 10 for our work.

### IMU-based Detection Window

For visibility and other practical reasons, traffic signals are placed high above the ground. As a result, traffic signals often appear only in the upper part of a captured video frame. As shown in Figure 4.6, the lower half of the image captures the road and other low-lying objects, whereas the traffic signals are located in the upper half. The part of the image where the traffic signal can be located depends not only on the orientation of the windshield-mounted smartphone, but also on the distance from the traffic signal; the closer the phone to the traffic signal, the higher the signal appears in the image for a given device orientation.

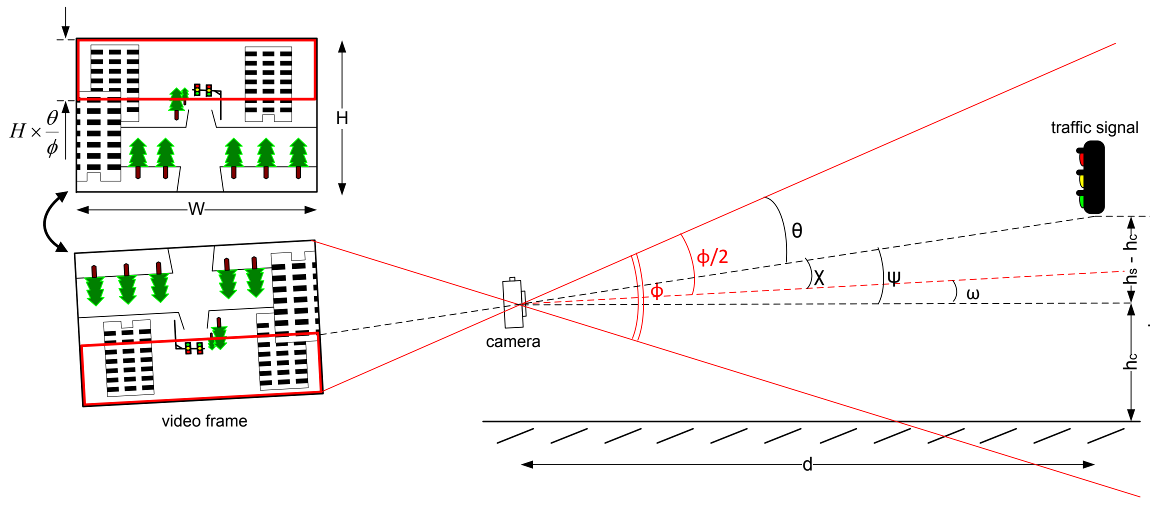


Figure 4.7: **Detection window calculation. $\phi$ is the smartphone camera's vertical angle of view. $d$ is the distance of the smartphone device from the traffic signal and is calculated based on the GPS location of the device. $h_s$ and $h_c$ are relatively fixed and are the heights of the traffic signal and camera, respectively. $\omega$ is the roll angle of the camera and is calculated by the accelerometer- and gyro-based IMU.**

SignalGuru leverages information from the smartphones' inertial sensors to narrow its *detection window*, *i.e.,* the part of the image where traffic signals are expected to appear. More specifically, SignalGuru uses information from the accelerometer and gyro-based IMU of the smartphone to infer its orientation (roll angle) and information from its GPS device to calculate distance from the traffic signal.

With this information, the size of the detection window can be easily calculated analytically. As shown in Figure 4.7, the traffic signal is located within the angle $\theta$. Hence, if $\phi$ is the camera's vertical angle of view, then the part of the image that needs to get processed is only the upper $\theta/\phi$ fraction[4]. For iPhone 3GS and iPhone 4 devices, $\phi = 34.6\,^\circ$ and $47.5\,^\circ$, respectively. The angle $\theta$ is calculated as: $\theta = \phi/2 - \chi$ where $\chi = \psi - \omega$ and $\psi = arctan(h_s - h_c)/d$ (see Figure 4.7). The height of the detection window is then cropped to $H \times \theta/\phi$. The IMU-based detection window is shown with a red bounding box in Figures 4.6 and 4.7.

[4]Using the small angle approximation, tan$x \approx x$.

The IMU-based detection window scheme enables SignalGuru to ignore a large portion of a captured frame that can have nothing but noise, providing twofold benefits: First, the image processing time is almost halved, and second the traffic signal detection is significantly improved. The benefits of this scheme are evaluated in Section 4.7.2.

#### Variable Ambient Light Conditions

Ambient light conditions significantly affect the quality of captured still images and video frames. The amount of ambient light depends on both the time of the day and the prevailing weather conditions (sunny vs. cloudy). Smartphone cameras automatically and continuously adjust their camera exposure setting to better capture the target scene. Nevertheless, we found that traffic signals are often not captured well with their bulbs appearing either too dark (underexposed) or completely white (overexposed). As a result, the detection module would perform very poorly in some cases.

Traffic signals, however, have a fixed[5] luminous intensity. We leverage this by adjusting and locking the camera exposure time to the fixed intensity of traffic signals. This eliminates the sensitivity of traffic signal detection to time of day or weather. The camera exposure time is automatically adjusted by pressing the "Adjust Exposure" button and pointing the camera to a traffic signal. Then by pressing the "Lock Exposure" button, the setting is recorded and locked, obviating the need for further adjustments.

### 4.5.2 Transition Filtering Module

The raw detection of traffic signals and their color transitions (R$\rightarrow$G) given by the detection module is fairly noisy. In our Singapore deployment, in 65% of the cases

[5]LED traffic signals have fixed luminous intensity. Older incandescent traffic signals do not, but are quickly becoming obsolete. Both Cambridge and Singapore use LED traffic signals.

that a vehicle is waiting at a red traffic signal, it reports a false positive transition, *i.e.,* a transition that did not actually occur. Typically, the image detection module was detecting the actual red light and then happened to misdetect some arbitrary object for a green light. Note that vehicles were waiting at the intersection for $48s$, on average, capturing and processing perhaps dozens of video frames. If not handled properly, a single false green light detection would be enough to erroneously generate a transition report. Similarly, if a vehicle happens to misdetect an arbitrary object for a red light in between detections of the actual green light, a false transition would be reported.

While, ideally, we would like to detect and report all R→G transitions witnessed (no false negatives), it is even more critical to avoid false positives (reports of transitions that never happened), because false positives pollute the prediction scheme. Therefore, we filter R→G transitions using a two-stage filter: A Low Pass Filter (LPF) in the first stage and a *colocation* filter in the second stage.

## Low Pass Filter (LPF)

According to our findings from our Singapore deployment, in 88% of the cases, false positive detections occur over a single frame and do not spread over multiple consecutive frames. As a result, most false transitions have one of the following three patterns with the false detection marked in bold:

1) R→ ...→R→**G**→R→...→R

2) G→ ...→G→**R**→G→...→G

3) NS→...→NS→**R**→**G**→NS→...→NS

The first (most common) pattern occurs when the vehicle is waiting at the red light it correctly detects, then at a specific instance it misdetects a passing object (*e.g.,* design on a bus crossing the intersection) for a green traffic light. The second pattern occurs when the vehicle misdetects an arbitrary object for a red light in between detections of the actual green light. Finally, the third pattern occurs when the view of the vehicle is obstructed and there is no traffic signal in sight. However, at some point, it misdetects an arbitrary object for a red light and right after that a different object for a green light. This pattern is the least common.

The LPF filters out such "spikes" or anomalies across multiple traffic signal transitions by adding some hysteresis. The LPF classifies only transitions that have the R$\rightarrow$ R$\rightarrow$G$\rightarrow$G pattern as valid, *i.e.,* at least two red status reports followed by at least two green status reports. As our results in Section 4.7.3 show, the LPF filters the vast majority of false positive transitions at the cost of creating only a small number of false negatives (actual transitions removed by the filter).

### Colocation Filter

A distinctive feature of traffic signals, as opposed to other objects with similar colors and shape, is that the red and the green bulb are contained in the same black box, that is, they are *colocated.* SignalGuru's filtering module leverages this by checking whether detected red and green bulbs are colocated before accepting a transition as valid. More specifically, the colocation filter checks whether the green bulb that was just detected is close to the red bulb detected in the previous frame. Note that the accumulator of the Hough transform pinpoints the location of the traffic signal candidates.

Given that SignalGuru can capture and process video frames every $2s$, the average delay between the light turning green and SignalGuru capturing this event in its next video frame is $1s$. In $1s$ or even $2s$ that is the maximum possible green light capture delay, a vehicle will not have accelerated and moved significantly. Hence, there is no need to compensate for vehicle movement.

However, as exemplified by the photo in Figure 4.3, many intersections have two or more traffic signals for the same phase, or for the same direction of traffic. Therefore, the red and green bulbs may be detected on different traffic signals across the two frames. To tackle this, before the colocation filter rejects a transition as invalid, it invokes the detection module to check whether there exists, in the current frame, a green bulb that is collocated with the red bulb of the previous frame. In this case, the detection window covers only a very small area around the red traffic signal of the previous frame, and thus incurs negligible computational overhead.

As shown in Section 4.7.3, the colocation filter effectively filters out false positive transitions at the cost of a small increase in false negatives. Together, the LPF and the colocation filter form a very robust two-stage filter.

### 4.5.3 Collaboration Module

SignalGuru depends on the grassroots collaboration among the participating nodes (smartphones). A node is limited by its field of vision, and does not have all the information it needs in order to predict the schedule of the traffic signals ahead. Typically, a node needs information about a traffic signal well before the signal comes into the node's camera field of view. As a result, the predictions of a node depend on data sensed by the node's predecessors in a given intersection.

For the prediction of traffic-adaptive traffic signals, collaboration is even more critical. As we explain in Section 4.5.4, in order to predict traffic-adaptive traffic signals, information from all phases (intersecting roads) of an intersection is needed. Furthermore, Section 4.7.4 shows how more collaborating nodes and more traffic signal history can improve the prediction accuracy for the challenging traffic-adaptive traffic signals of Singapore.

The collaboration module allows participating SignalGuru nodes to opportunistically exchange their traffic signal information (timestamped R$\rightarrow$G transitions) by periodically (every two seconds) broadcasting UDP packets in 802.11 ad-hoc mode. A SignalGuru node exchanges not only the data it has sensed on its own, but also the data it has opportunistically collected so far. Only data about the traffic signal transitions of the last five cycles is exchanged. We found that using a longer history of data does not significantly improve the traffic signal prediction accuracy.

In order to be able to predict the schedule of the traffic signals ahead (Section 4.5.4), nodes need either the database of the traffic signal settings (for pre-timed traffic signals) or the SVR prediction models (for traffic-adaptive signals). This information is passed to a node along with the sensed transition data before the node approaches the corresponding traffic signals. However, it is likely that this node will have also crossed the traffic signal ahead in the recent past (*e.g.,* yesterday due to daily commute). In this case, the sizeable (62 KB) SVR prediction models do not need to be sent again as they are relatively static (Section 4.7.4). The settings for a pre-timed traffic signal can be encoded in just a few bytes and thus sending them again incurs negligible overhead.

More specifically, when running atop RegReS, SignalGuru service carriers will need to exchange the following three types of data:

1. *Service Advertisement (SA) data*: SignalGuru carriers broadcast periodically (every $T$=2sec) SA packets. RegReS uses SA packets in order to be able to estimate the current density of service carriers and decide when it is necessary to spawn new carriers. SA packets also allow for discovery of service carriers by other nodes that are interested in receiving traffic signal schedule information, as well as discovery of potential carrier nodes if spawning is necessary. Last, the service version field of SA packets allows existing carriers to detect if they have different service data (versions). In this case, they will request the newer data from the other service carriers.
2. *Carrier Spawn (CS) data*: An existing SignalGuru carrier passes to a newly spawned carrier the information about all the traffic signal transitions that got collaboratively detected within the specific region and over the last five cycles. More specifically, for each detected transition, the timestamps of the last red

and first green color detections are sent. For traffic-adaptive traffic signals, the SVR prediction model for each regional traffic signal (phase) is passed to newly spawned carriers as well, if necessary. CS data are also exchanged to update an existing carrier that has older service data (version).

3. *Service Update (SU) data*: When a new traffic signal transition is detected, its information is epidemically forwarded to all other regional carriers.

The amount of traffic signal information[6] that SignalGuru nodes gather and exchange can be constrained by tiling a geographic area into regions and having SignalGuru nodes maintain and exchange data that belong only to their current region. Different ways to tile an area into regions along with their respective network overheads are evaluated in Section 4.7.5.

### 4.5.4 Prediction Module

Two main categories of traffic signals exist: pre-timed and traffic-adaptive traffic signals. Since their operation is very different, SignalGuru uses different prediction schemes for each category.

#### Pre-timed Traffic Signals

SignalGuru's prediction module maintains a database of the traffic signal settings. As described in Section 4.4, pre-timed traffic signals have fixed pre-programmed settings for their different modes (a.m./p.m. peak, off-peak, Saturday peak). Traffic signal settings can be acquired from city transportation authorities. In case they are not available, the settings (phase lengths) can be measured collaboratively, as described in the next section. This means that SignalGuru knows how long each phase lasts. The challenge remains to accurately synchronize SignalGuru's clock with the time of

[6]Sensed traffic signal transitions, database of traffic signal settings and SVR prediction models for pre-timed and traffic-adaptive traffic signals, respectively.

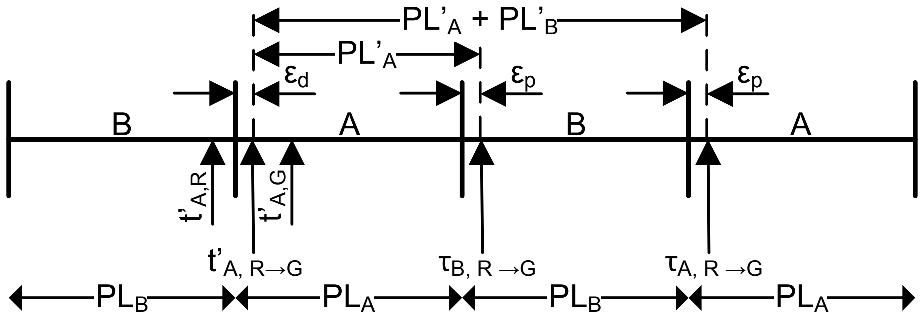


Figure 4.8: **Timeline of traffic signals operation and SignalGuru's detections and predictions for a simple intersection with two phases (A and B). The letters on the timeline denote for which of the two phases the light is green. The timestamps of actual, detected and predicted phase transitions are also marked with $t$, $t'$ and $\tau$, respectively. PL$_A$ is the actual length of phase A and PL'$_A$ its predicted value. $\epsilon_d$ and $\epsilon_p$ are the color transition detection and prediction errors, respectively.**

phase transition of a traffic signal. Once this is achieved, SignalGuru can very easily predict when the traffic signal will switch again to green, yellow or red.

Clock synchronization is achieved by capturing a color transition, *e.g.,* R→G. Figure 4.8 shows a timeline of events. If the timestamps of the last red and first green color detections for phase A are $t'_{A,R}$ and $t'_{A,G}$, respectively, then the detected transition time is $t'_{A,R\to G} = (t'_{A,R} + t'_{A,G})/2$. Clock synchronization needs to be reestablished after a false R→G detection and every time the traffic signal changes mode of operation or recovers from an operational failure.

The time the traffic signal will switch to red for phase A (and green for phase B) can be predicted by adding the predicted[7] length of phase A ($PL'_A$) to $t'_{A,R\to G}$ as $\tau_{B,R\to G} = t'_{A,R\to G} + PL'_A$. Since this intersection has only two phases, phase A will follow after phase B; as phases are scheduled in a predictable, round-robin way. By adding ($PL'_B$) to $\tau_{B,R\to G}$, we get the next R→G transition for phase A, and so on.

[7]For pre-timed traffic signals, the predicted phase length is the value looked up in the traffic signal settings database.

**Traffic-adaptive Traffic Signals**

The Singapore GLIDE (SCATS) system constitutes one of the most sophisticated and dynamic traffic-adaptive traffic signal control systems. As described in Section 4.4, SCATS measures the saturation of the intersections' phases and adjusts their phase lengths at every cycle. Phase lengths change when SCATS changes the value of the cycle length or the fraction of the cycle length that gets allocated to them. SCATS may choose to change both settings at the same time. Phases are still scheduled in a deterministic round-robin manner.

SignalGuru predicts future transitions (*e.g.,* when the signal ahead will turn green) by detecting past transitions and predicting the length of the current or next phases. The key difference from the prediction of pre-timed traffic signals lies in the prediction of the *phase length*, as opposed to looking it up from a database.

SignalGuru predicts the length of a phase by measuring and collaboratively collecting the prior traffic signal transition history, and feeding it to an SVR [23] prediction model. In Section 4.7.4, we evaluate the prediction performance of different Prediction Schemes (PSs) by training the SVR with different sets of features:

- PS1: The next length of a given phase, *e.g.,* A, is predicted based on the history of the same phase, *i.e.,* the next length of phase A is predicted based on the lengths of the five previous phases of A. We found that further increasing the length of the history does not yield any benefits. Similarly, SCATS uses only loop-detector measurements performed over the last five cycles to determine the next traffic signal settings.
- PS2: Like PS1, but the length of the preceding phases of the same cycle is also provided. This means that when trying to predict the length of phase C, the lengths of preceding phases A and B are also fed to the SVR model. As our results show, this information improves significantly the performance of the prediction module. The reason is that changes to a given cycle's phase lengths

are correlated when SCATS changes just the cycle length setting, instead of the phase length setting.

- PS3: Like PS2, except that information for the past five cycle lengths is also factored in.
- PS4: This is a theoretical prediction scheme. We assume the existence of loop detector saturation information in addition to PS3. Saturation values over the past five cycles are fed to the SVR model. Note that traffic (vehicle speed) estimation is not a good proxy for the unavailable loop detector measurements. Average vehicle speed does not always correlate well with the saturation measured by SCATS's loop detectors; a specific phase, despite the fact that vehicles are moving fast, may be highly saturated (with dense flow).

One-week-long history of data is enough to train the SVR model, as our results show. Furthermore, the SVR model does not need to get continuously re-trained. Re-training the model every four to eight months is frequent enough in order to keep the prediction errors small.

In order for SignalGuru to be able to use any of the first three feasible prediction schemes, the lengths of the past phases need to be measured. While it is easy for SignalGuru to detect the $R \rightarrow G$ transition for the beginning of a phase, as explained in Section 4.5, it is very hard to detect the $G \rightarrow Y$ transition for the end of the phase. To remedy that, *collaboration* across nodes waiting at the different traffic signals of the same intersection is leveraged; the $G \rightarrow Y$ transition of a given phase is inferred by the $R \rightarrow G$ transition of the successor phase that was detected by nodes waiting at the traffic signal of the successor phase. For example, the fact that the light turned green for phase B at time $t$ means that it turned yellow for phase A at time $t$ minus the *clearance interval*. The clearance interval is a fixed setting and is the amount of time a phase is yellow plus the amount of time all phases are red before the next one turns green. As its name denotes, it gives the previous phase enough time to clear before the next conflicting phase starts.

## 4.6 Methodology

This section describes the two deployments that we performed in Cambridge and Singapore to evaluate SignalGuru.

### 4.6.1 Cambridge Deployment

As mapped in Figure 4.9, our November 2010 deployment in Cambridge targeted three consecutive intersections on Massachusetts Avenue. We used five vehicles with iPhones mounted on their windshields and asked the drivers to follow the route shown for $\sim$3 hours. Given the small number of available vehicles, all vehicles were used for SignalGuru and collaborated with an epidemic broadcast-based scheme, as opposed to using RegReS's selective service activation. Note that the opportunity for node encounters (within ad-hoc wireless range) was small, as all the vehicles followed the same route so they were rarely in range of each other. To rectify this, an extra iPhone device was held by a pedestrian participant located at the intersection of Massachusetts Avenue and Landsdowne Street. This SignalGuru device served as an ad-hoc data relay node, facilitating data exchange between the windshield-mounted iPhone nodes. Only the collaboration module was active on the relay node. The experiment took place between 1:20pm - 4:30pm. At 3:00pm, the traffic signals changed operation mode from off-peak to afternoon peak.

### 4.6.2 Singapore Deployment (Bugis Downtown Area)

Our other deployment was in Singapore in August 2010. Unlike Cambridge, the Singapore deployment tests SignalGuru on traffic-adaptive traffic signals. To measure

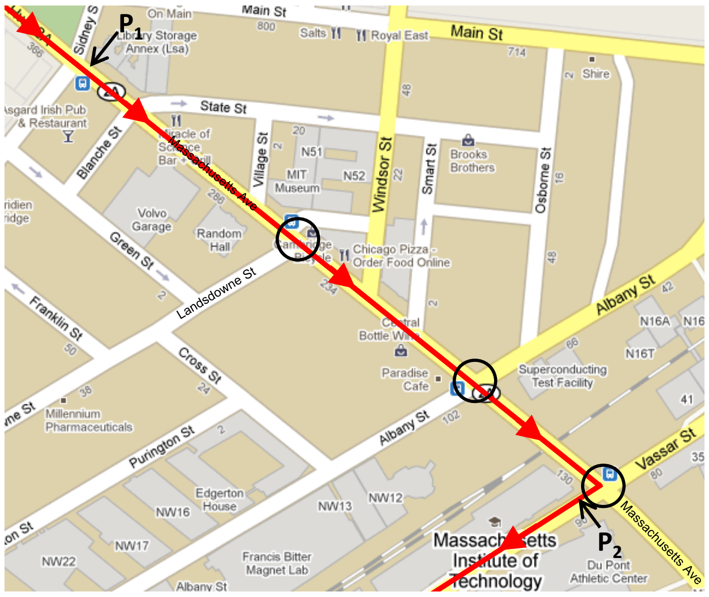


Figure 4.9: **Route of vehicles in Cambridge deployment. The targeted intersections are marked with circles. P$_1$ and P$_2$ are the start and end points, respectively, for our GLOSA experiment trip.**

phase lengths and predict the schedule of traffic-adaptive traffic signals, SignalGuru needs to monitor all phases of an intersection, *i.e.,* orthogonal directions of a traffic intersection. Hence, in this deployment, we had two sets of vehicles following the two distinct routes shown in Figure 4.10. In this way, both phases of the intersection (Bras Basah and North Bridge Road in Singapore's downtown) were sensed. Phase A corresponds to vehicles moving along Bras Basah Road and phase B to vehicles moving along North Bridge Road.

We used eight iPhone devices in total and mounted them on the windshields of taxis. Five devices were moving on the longer route of phase A and the other three on the shorter route of phase B. Similarly to our deployment in Cambridge, collaboration was enabled with an epidemic broadcast-based scheme and an extra iPhone device was used as a relay node. In this case, the relay node was also recording the ground truth[8], *i.e.,* when the traffic signals status transitioned. Ground truth information

[8]In our Cambridge deployment, since the schedule of the signals is fixed, it can be easily inferred from the images logged by the windshield-mount iPhones. Hence, there was no need to record the ground truth with an extra iPhone device.

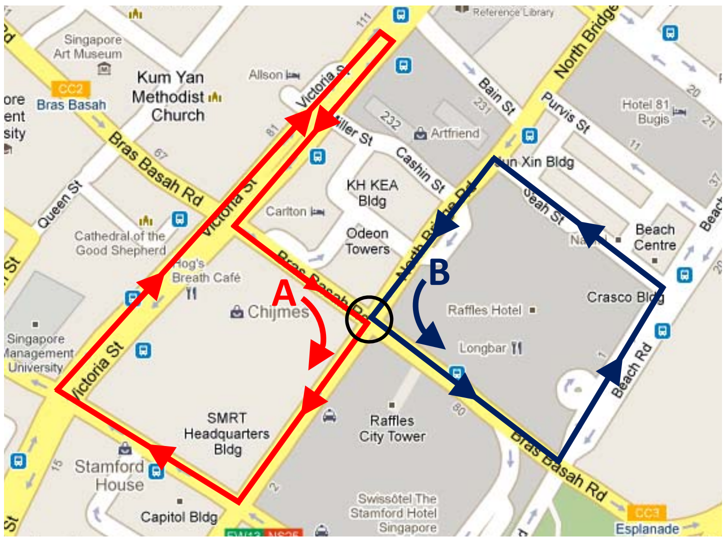


Figure 4.10: **The two distinct routes of taxis in Singapore deployment in the Bugis downtown area. Routes A and B correspond to phases A and B of the targeted intersection, respectively. The targeted intersection is marked with a circle.**

was only used for offline evaluation of SignalGuru's accuracy thereafter. It was not shared with other participating nodes. The experiment took place from 11:02am - 11:31am (∼30min).

## 4.7 SignalGuru Evaluation

Here, we evaluate the performance of each of SignalGuru's modules before evaluating its overall performance in two deployments in Cambridge and Singapore. We also performed a large-scale analysis for SignalGuru's prediction accuracy based on the data we collected from Singapore's Land Transport Authority.

### 4.7.1 Traffic Signal Detection

We evaluate the performance of SignalGuru's detection module for our two deployments. In Figure 4.11, we show both the percentage of false negatives (traffic signals that did not get detected) and the percentage of false positives (arbitrary objects confused

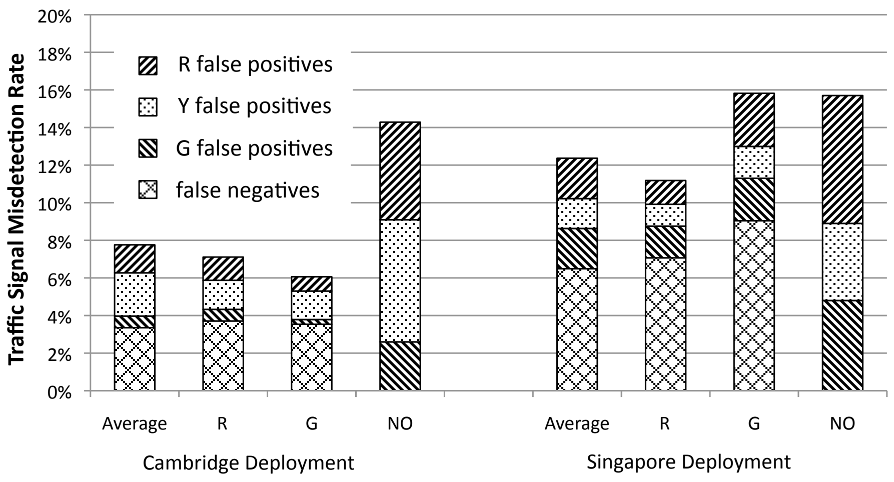


Figure 4.11: **Traffic signal detection module evaluation. R/Y/G/NO stands for video frames where the traffic signal is actually Red/Yellow/Green or non-existent. A false negative is when the module fails to detect the existing traffic signal. A false positive is when the module confuses an arbitrary object for a traffic signal of a specific R/Y/G status. We omitted "Y" results as there were very few such frames and hence the detection results are not statistically important.**

for traffic signals of a specific color). Results are averaged over 5959 frames and 1352 frames for the Cambridge and Singapore deployments, respectively. The average misdetection rate that includes both false negatives and false positives was 7.8% for Cambridge and 12.4% for Singapore deployment. In other words, SignalGuru's detection module correctly detected the existence (or the lack) of a traffic signal in 92.2% and 87.6% of the cases in Cambridge and Singapore, respectively. Note that most (>70%) video frames are captured while vehicles are waiting at the red light. Hence, the average (mis)detection rate is strongly biased by the results for "R", *i.e.,* frames with a red traffic signal.

As Figure 4.11 shows, the detection module is particularly more likely to report a false positive when there is no traffic signal in sight. When a traffic signal is captured in the video frame, the actual traffic signal will normally get the most votes in the Hough transform's accumulator and a valid detection will be recorded. If there is no traffic signal in sight, the detection module will try to find the best possible candidate object that most resembles a traffic signal in terms of its color, shape and enclosing black box, which can trigger more false positives.

Furthermore, the ratio of false positives of different colors differs significantly across the two deployments. For example in Cambridge, yellow light false positives are more common than in Singapore, where there are more green light false positives. This is because of the prevailing ambient light conditions and the object composition of the environment at the targeted intersections. In Singapore, there were many more trees and also a black electronic message board with green letters, whereas in Cambridge, the sun was setting, giving a strong yellow glare to several objects (*e.g.,* road signs, vehicles, buildings, *etc.*).

Another interesting observation is that the number of false negatives (missed traffic signal detections) is almost double in the Singapore deployment, as compared to the Cambridge deployment. The reason lies in the traffic signal bulbs used in each city. Singapore's LED bulbs are exposed, whereas Cambridge's are covered by a refraction lens. The LED traffic signal bulbs consist of an array of smaller LEDs that is refreshed in columns at a relatively low frequency. The refresh frequency is high enough to be invisible to the human eye but low enough to be detectable by a camera when there is no refraction lens covering the bulb. In Singapore, the camera would thus sometimes capture the bulbs with dark stripes (columns) of LEDs that have not got refreshed, reducing the probability of a successful traffic signal detection.

### 4.7.2 IMU-based Detection Window

In this section, we evaluate the benefits that the IMU-based detection window offers. The iPhones were oriented horizontally, as shown in Figure 4.3. The lower line of the detection window will thus be horizontal and across the center of the image when the vehicle is at a distance of $\sim 50m$ from the intersection. The results, for when the IMU-based detection window was activated/deactivated, were acquired by

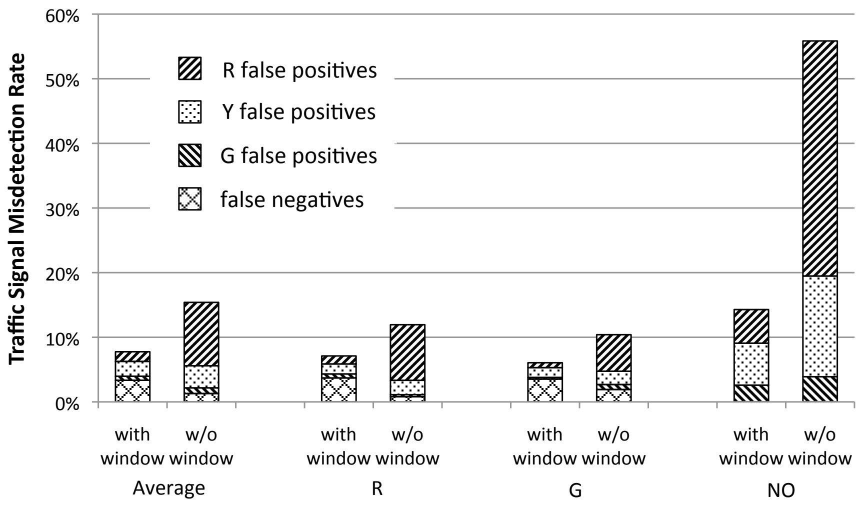


Figure 4.12: **IMU-based detection window scheme evaluation for Cambridge deployment. The IMU-based detection window scheme almost halves the rate of misdetections.**

online/offline traffic signal detection. The offline detection was based on the same video frames that were logged and processed by the iPhone devices online.

The comparisons in Figure 4.12 show that the IMU-based detection window almost halves the average misdetection rate reducing it from 15.4% to 7.8%. Above all, the IMU-based detection window significantly reduces the number of red false positives; when the detection window scheme is not used and the whole video frame is processed, the detection module often confuses vehicles' rear stop lights for red traffic signal bulbs.

On the other hand, the IMU-based detection window scheme increases the number of false negatives when the traffic signal is red; when a vehicle is decelerating abruptly to stop at the red light, the IMU miscalculates the device's orientation. As a result, the detection window is also miscalculated, becoming so small that the traffic signal is excluded. Nevertheless, the effects of abrupt decelerations are only transient and a car is soon able to detect the traffic signal ahead.

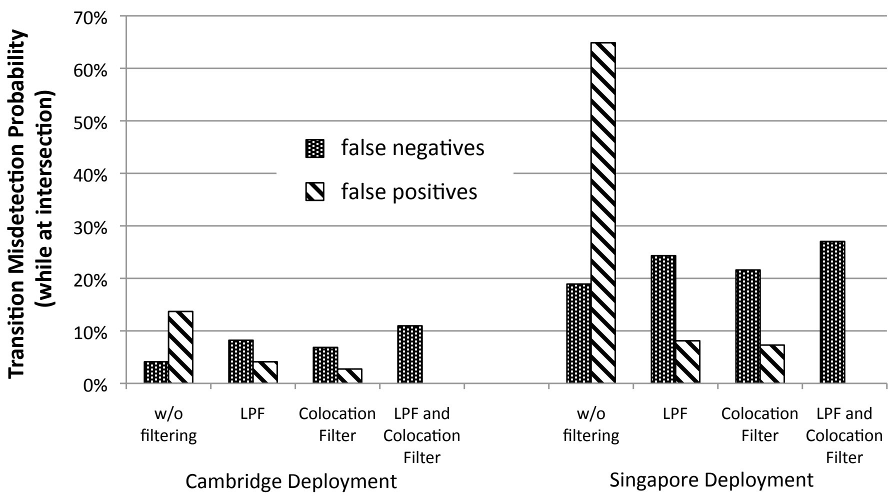


Figure 4.13: **Transition filtering module evaluation. The transition filtering module removes false positive reports without significantly increasing the number of false negatives. The iPhone devices witnessed 219 and 37 traffic signal transitions in our Cambridge and Singapore deployments, respectively.**

Overall, since only a fraction of the video frame is processed, the IMU-based detection window scheme reduces the average processing time by 41% (from 1.73$s$ to 1.02$s$).

### 4.7.3 Transition Filtering

The performance of the transition filtering module is evaluated in terms of the number of false positives (invalid transitions) it manages to remove and the number of false negatives it creates (valid transitions erroneously removed).

As shown in Figure 4.13, the probability of (unfiltered) false positives in the Cambridge deployment is significantly smaller when compared to the Singapore deployment. This occurs for two reasons: First, the rate of false positive traffic signal detections is smaller in Cambridge. Second, the average waiting time at red traffic signals is only 19.7$s$ for Cambridge vs. 47.6$s$ for Singapore. As a result, the probability of a false positive transition detection during that waiting time is significantly lower.

While the LPF and colocation filters each significantly reduce the number of false positives, it is when both filters are applied in series that all false positives are removed in both deployments, with only a small increase in the number of false negatives. More specifically, the probability of false negatives increased by 6.8% for Cambridge and 8.1% for Singapore. Thus, the transition filtering module effectively compensates our lightweight but noisy traffic signal detection module.

### 4.7.4 Schedule Prediction

This section evaluates the accuracy of SignalGuru's traffic signal schedule predictions.

#### Cambridge Deployment

We evaluate the overall accuracy of SignalGuru's traffic signal schedule predictions for Cambridge's pre-timed traffic signals. As evaluation metric, we use the prediction mean absolute error; the absolute error between the predicted and the actual traffic signal phase transition time, averaged across the 211 predictions performed by the participating iPhone devices.

As shown in Figure 4.14, SignalGuru can predict the schedule of pre-timed traffic signals with an average error of only $0.66s$. Since SignalGuru uses a database for the settings of pre-timed traffic signals, the prediction error is solely caused by the error with which SignalGuru detects color (phase) transitions. When SignalGuru captures and processes video frames every $T{=}2s$, the transitions are theoretically detected with an error that has a maximum value of $\epsilon_{max}{=}T/2{=}1s$ and an expected value of $[\epsilon]{=}T/4{=}0.5s$. This is very close to the measured prediction error value of $0.66s$. Given this very small prediction error, our SignalGuru can effectively support the accuracy requirements of all applications described in Section 4.9.

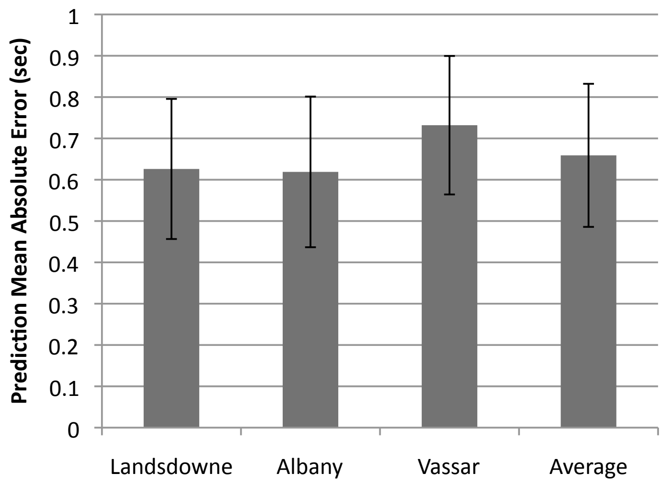


Figure 4.14: **Mean absolute error of SignalGuru's traffic signal schedule predictions for the three targeted intersections in Cambridge. The error bars show the standard deviation of the mean absolute error. The ground truth on the status of traffic signals was inferred by the images logged by the windshield-mounted iPhones with sub-300$ms$ accuracy.**

### Singapore Deployment

We evaluate the accuracy of SignalGuru's traffic signal schedule predictions for Singapore's traffic-adaptive traffic signals, using the prediction mean absolute error as the evaluation metric. The prediction module was configured to use the prediction scheme PS3, and was trained offline using a week's worth of data (June 1-7 2010) that we obtained from Singapore's Land Transport Authority (LTA).

As our results in Figure 4.15 show, SignalGuru can predict the time of the next color transition with an average error of 2.45$s$. The next color transition prediction error is broken down into an average absolute error of 0.60$s$ in detecting the current phase's start time (detection module error) and an average absolute error of 1.85$s$ in predicting the length of the current phase (prediction module error). The prediction error is due to both the inaccurate phase duration measurements that are fed into the SVR model and the prediction error of the SVR model, with the latter being the main contributor. The phase duration measurement error has a triangular probability

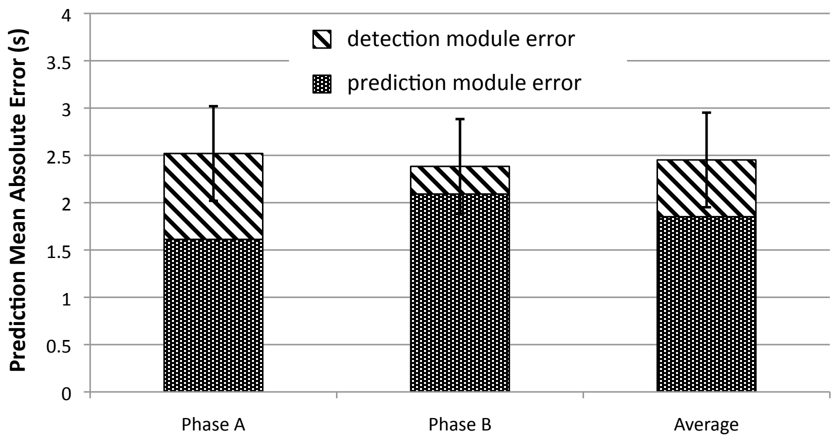


Figure 4.15: **Traffic signal schedule prediction evaluation for Singapore deployment. The ground truth was recorded every two seconds and the actual (ground truth) transition time for a phase** *e.g.,* **A was calculated as** $t'_{A,R \to G} = (t'_{A,R} + t'_{A,G})/2$**. The measurement error was thus 1**$s$ **(shown with error bars).**

density function[9] and the expected value for the phase duration measurement absolute error is only $[\epsilon_{duration}]$=$T/3$=$0.66s$. Results are averaged over 26 predictions. The schedule prediction accuracy for the two phases is comparable.

Without collaboration, SignalGuru would not be able to measure the past length of the phases for the purpose of feeding them into the SVR-based prediction scheme and predicting their future lengths. SignalGuru would have to predict the future phase length for a traffic-adaptive traffic signal using the same scheme that it uses for pre-timed signals, *i.e.,* by using a typical fixed value for it. Such a value could be, for example, the average length of the phase during the same hour of the previous day. In this case, the prediction error would have been $11.03s$ instead of $2.45s$ ($3.5\times$ higher). Collaboration is critical for accurate traffic signal schedule predictions, particularly for traffic-adaptive traffic signals.

[9]Assuming independent and uniformly distributed in [0, $T/2$] errors for the detection of the phase start and stop times.

**Singapore Large-scale Prediction Analysis**

In order to perform a large-scale evaluation of the performance of SignalGuru's prediction module across different traffic signals and intersections with different traffic patterns, we collected traffic signal operation logs from Singapore's Land Transport Authority. More specifically, we collected logs for 32 traffic signals (phases) in the Dover (suburban) area and for 20 traffic signals (phases) in the Bugis (downtown) area. The logs spanned over the two weeks of June 1-14, 2010 and contained more than 200,000 phase lengths for both Bugis and Dover traffic signals. We used the logs of the first week to train the different SVR-based prediction schemes, and the logs of the second week to test their performance. The training and testing sets were therefore not overlapping.

**Prediction Schemes Evaluation.** In Figure 4.16, we evaluate the performance of the different phase length prediction schemes for the traffic signals of Dover and Bugis. We also include the performance of a baseline scheme PS0 that uses the last measurement of a phase's length as the prediction for its future length. PS3 outperforms PS1 and PS2 and reduces the phase length prediction mean absolute error by 37% (from $3.06s$ to $1.92s$) for Bugis and by 26% (from $1.60s$ to $1.194s$) for Dover when compared to PS0.

As shown in Figure 4.16, the prediction mean absolute error for Dover traffic signals is half when compared to the error for Bugis traffic signals. However, note that the average phase length for Bugis is $47s$ whereas for Dover it is only $28s$. As a result, the relative errors (when compared to their own average phase length) are more comparable: 4.1% for Bugis and 4.3% for Dover.

Surprisingly, we found that the theoretical prediction scheme PS4, which assumes knowledge of loop-detector information, does not outperform PS3. We believe that this is because the effects of loop-detector measurements are already captured by SCATS

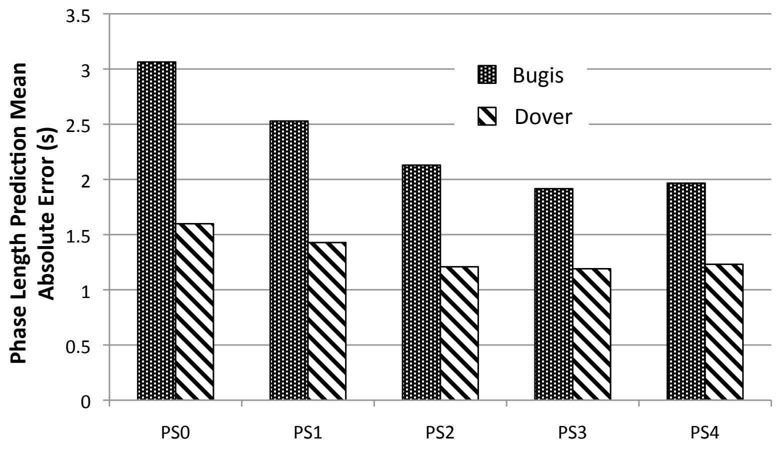


Figure 4.16: **Evaluation of the different prediction schemes for Bugis and Dover traffic signals.**

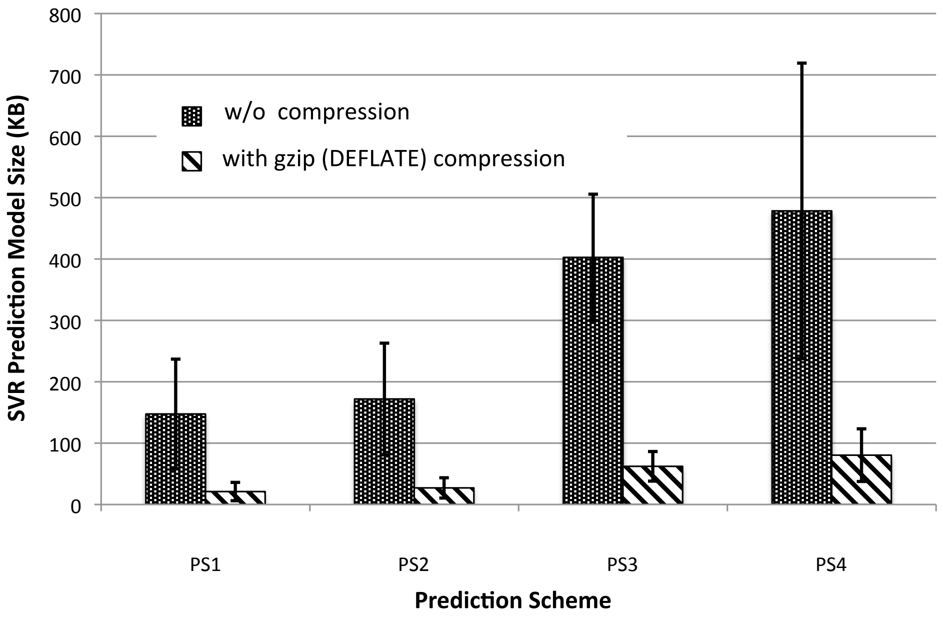


Figure 4.17: **Prediction model sizes for the different prediction schemes.**

in the history of the phase and cycle length settings that it chooses and SignalGuru measures them and uses them as prediction features for PS3.

As Figure 4.17 shows, the more complex the prediction scheme, the bigger the size of its model. However, after compression, the size of PS3's model used by our SignalGuru remains relatively small (67 KB).

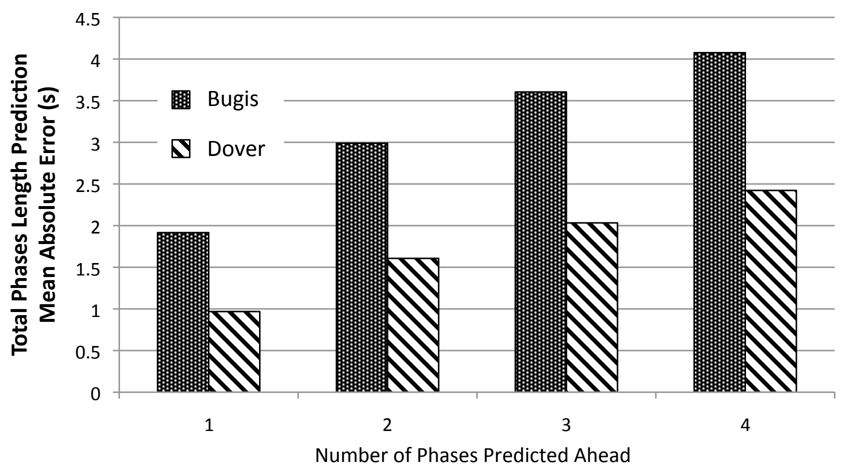


Figure 4.18: **Evaluation of SignalGuru's prediction scheme PS3 when predicting multiple phases ahead for Bugis and Dover traffic signals.**

**Increasing available lead-up time.** In order to increase the available lead-up time beyond the length of a single phase[10], SignalGuru needs to predict multiple phases ahead. For traffic-adaptive traffic signals, the prediction error increases as SignalGuru tries to predict multiple phases ahead. For pre-timed traffic signals, for which the phase lengths are fixed and known, the prediction error only depends on the ability of SignalGuru to synchronize with the traffic signal (by detecting a color transition as accurately as possible) and thus the lead-up time is arbitrarily long so long as it is within the same traffic mode.

Figure 4.18 shows the error of the prediction module, when it predicts the lengths of multiple phases ahead. The prediction error increases sublinearly as the number of predicted phases increases. However, even when predicting four phases ahead, the total prediction error for all phase lengths is only $4.1s$ (8.7%) and $2.4s$ (5.2%) for Bugis and Dover traffic signals, respectively. Given that wireless 802.11g broadcasts KB data over several hops in $< 1s$, the average available lead-up times for Bugis and Dover are $187s$ and $114s$, respectively. The percentage of available data (% transitions detected) in our Singapore deployment was 81%.

[10]Predicting a single phase in advance suffices for all proposed applications except for the TSAN application.

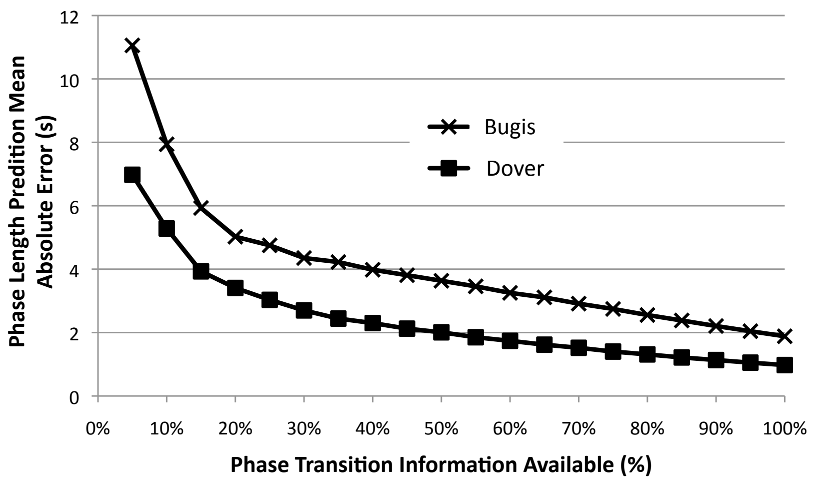


Figure 4.19: **Phase length prediction accuracy for Bugis and Dover traffic signals as the percentage of the available traffic signal transition data varies.**

As this analysis shows, SignalGuru can predict accurately the schedule of traffic-adaptive traffic signals regardless of their location, *e.g.,* suburban or downtown. Furthermore, their schedule can be predicted multiple phases in advance with small errors, enabling all the novel applications mentioned in Section 4.9 for traffic-adaptive traffic signals.

**Collaboration Benefits.** Figure 4.19 shows how the accuracy of phase length predictions depends on data availability, *i.e.,* the percentage of traffic signal transitions that are detected and made available (through collaborative sensing and sharing). Where the phase length cannot be determined (because no SignalGuru node detected its start or end), we used the previously predicted phase length. The more transition data are available (higher degree of collaboration), the better SignalGuru's prediction accuracy. When data availability drops below 25% for Bugis and 28% for Dover, relative prediction errors degrade to $>$10%. As a result, SignalGuru can no longer meet the requirements of the described applications. Collaboration is thus critical to ensure high-quality predictions.

**SVR re-train frequency.** We evaluate how well the SVR model that was trained using the data of June 1-7, 2010 can predict the schedule of the traffic signals after one

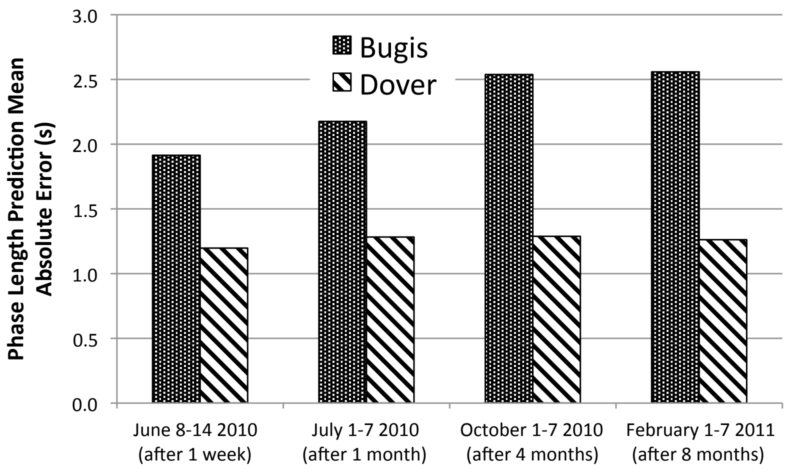


Figure 4.20: **Prediction model performance over time. The prediction performance of the SVR model that was trained with the data of June 1-7 2010 is evaluated for the weeks of June 8-14 2010, July 1-7 2010, October 1-7 2010 and February 1-7 2011.**

week (June 8-14, 2010), one month (July 1-7, 2010), four months (October 1-7, 2010) and eight months (February 1-7, 2011). As shown in Figure 4.20, the SVR model can make accurate predictions even after eight months for both Dover and Bugis. More specifically, the error for Dover traffic signals does not significantly increase over time. In contrast, for Bugis traffic signals, the prediction error increases by 33% (from 1.9$s$ to 2.6$s$) after eight months. Singapore's Land Transport Authority (LTA) engineers manually perform changes to the traffic signal settings (*e.g.,* phase programs) over time in an attempt to better optimize the traffic signals operation in the busy Singapore downtown area. As a result, SingalGuru's prediction ability degrades over time for Bugis, and the SVR model needs to get retrained every few months in order to keep prediction errors low.

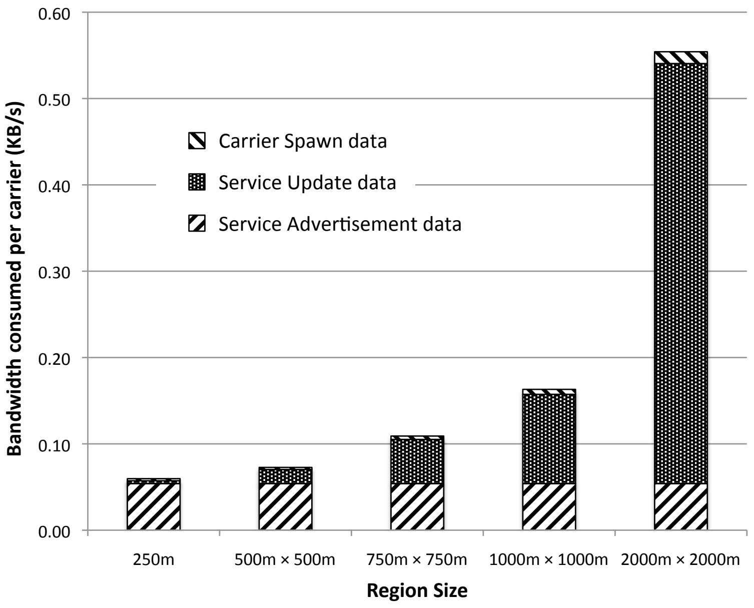
### 4.7.5 Service Overhead

In this section, we evaluate the resources that SignalGuru requires in terms of communication and computation.

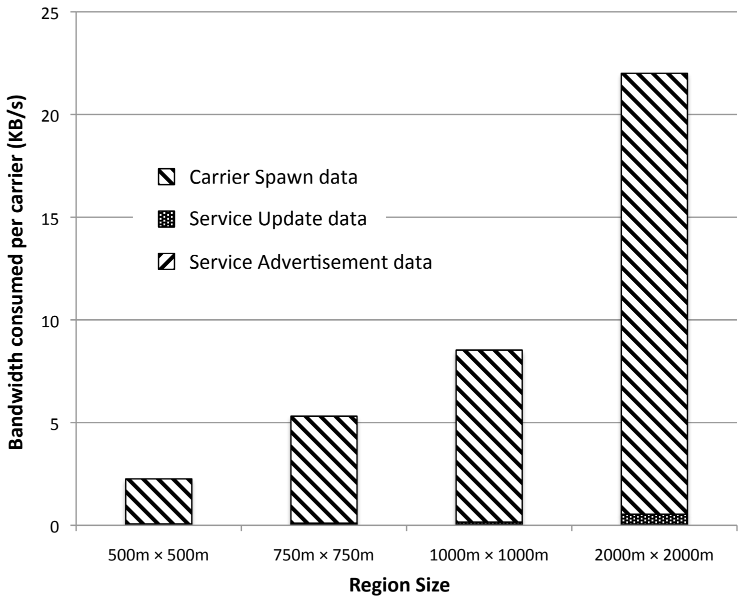
### Communication Resources

To calculate the communication overhead of a widely-deployed SignalGuru service, we assumed an implementation on top of RegReS that allows for selective service activation. The size of the different packets was calculated assuming that nodes maintain a history of only the last five transitions for each traffic signal phase and that each intersection has an average of 2.5 phases, as in Singapore's Bugis area. For traffic-adaptive traffic signals, the prediction scheme PS3 was used assuming a model size of 67 KB. Again, as shown in Figure 4.17, this is the average size of PS3 models for traffic signals in Bugis. A density of four SignalGuru carriers per 250-meter-long road segment was also assumed, with vehicles moving at an average speed of $10m/s$ and stopping at only 50% of the traffic signals, on average. With only half of the vehicles stopping at the traffic signal, on average, a density of four carriers per 250-meter-long road segment yields a high probability of having at least one vehicle detect the R$\rightarrow$G transition of the traffic signal. Given the small size of the regions, we also assume for simplicity that vehicles move only straight (no turns within the region). The average red light waiting time was calculated assuming that vehicles are equally likely to arrive at any time within the red light duration and assuming an average phase length of $47s$. This is the average phase length of traffic signals in Singapore's downtown area (Bugis).

In Figures 4.21(a) and (b), we show the bandwidth consumed per carrier for the different region sizes illustrated in Figures 4.22(a)-(c) for pre-timed and traffic-adaptive traffic signals, respectively. The amount of information exchanged depends greatly on the size of the regions into which an area is tiled. The bigger the region, the greater the number of included signalized intersections and, hence, the greater the amount of information exchanged between SignalGuru service carriers. However, the larger the region, the longer the possible available lead-up distance and time.



(a)



(b)

Figure 4.21: **Bandwidth required per carrier for different region sizes (Figures 4.22(a)-4.22(c)) of pre-timed (a) and traffic-adaptive (b) traffic signals. For traffic-adaptive traffic signals the sizeable SVR prediction models need to be included in the spawn packets, significantly increasing the required bandwidth. SignalGuru includes compressed PS3 models in its spawn packets.**

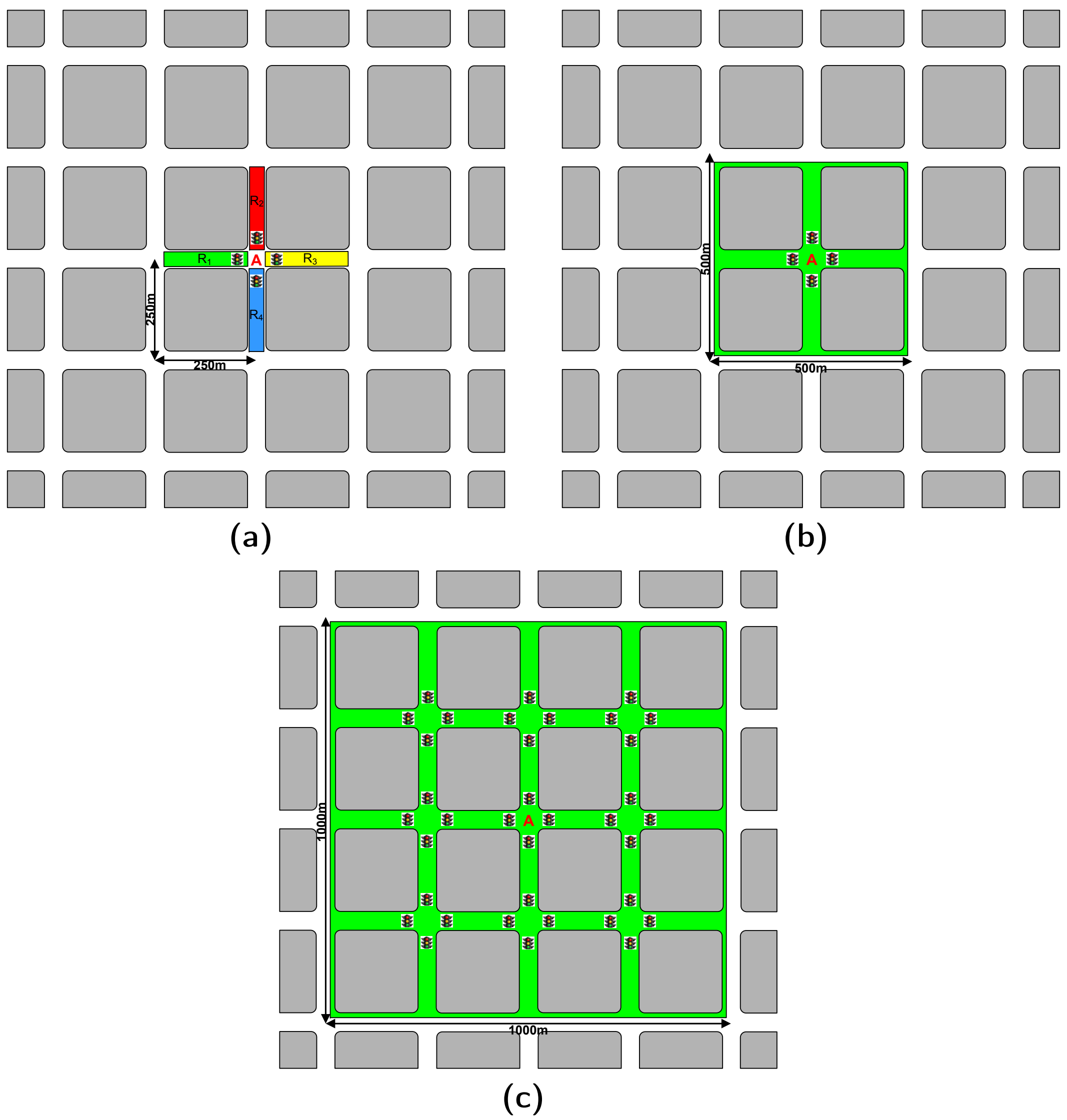


Figure 4.22: **Different region sizes for traffic signals of intersection "A". The traffic signals covered by the corresponding regions are shown. The "250m" region (a) consists of the single road segment before the traffic signal. Four such regions are needed to cover the whole intersection. The 500m x 500m (b) and 1000m x 1000m (c) regions cover one and nine intersections, respectively.**

As shown in Figure 4.21(a), for pre-timed traffic signals the per-carrier bandwidth consumption is dominated by SA and SU data. The bandwidth for SA packets is fixed, whereas the bandwidth for SU data increases as the region size increases. The bigger the region, the higher the number of included signalized intersections, and the larger the amount of detected transition information received and forwarded by each carrier. The bandwidth for CS data is small due to their small size and the relatively low frequency of SignalGuru service carrier spawns.

The same trends exist for the bandwidth of SA and SU data of traffic-adaptive traffic signals. However, for traffic-adaptive traffic signals, the CS data are significantly larger because of the sizeable SVR prediction models that need to be passed to newly spawned carriers. As a result, CS data, by far, dominate the communication bandwidth. For the same reason, the required bandwidth for traffic-adaptive traffic signals is 30×-40× higher as compared to the bandwidth for pre-timed traffic signals.

Overall, the communication resources required to collaboratively support SignalGuru atop the RegReS middleware is limited. Even for traffic-adaptive traffic signals, the required communication bandwidth per carrier is less than 22 $KB/s$. Furthermore, thanks to RegReS, which judiciously spawns only as many carriers as needed, the required aggregate communication resources for SignalGuru across a region are kept at bay.

**Computational Resources**

In this section, we present the computational (CPU, memory) resources that SignalGuru consumes. We profiled SignalGuru across one hour using Xcode's CPU Sampler performance tool. During the profiling, the iPhone 3GS device was facing the first intersection of Figure 4.9 at a distance of $\sim 30m$ from the traffic signal. SignalGuru was configured to process a new frame every 2 seconds using the IMU-based detection window scheme. The GPS and IMU modules were thus activated.

In Table 4.1, we show the computation time for the different components of SignalGuru. About 67% of the application CPU time is spent on traffic signal detection. This corresponds to 51% of system CPU time. The logging of images takes up 22% of the application CPU time[11]. For pre-timed traffic signals, the traffic

[11]Image logging was necessary in our experiments so that we can test whether the traffic signal detection was successful or not. In a real system, image logging would be disabled.

Table 4.1: **Computation resources required by SignalGuru's different modules. The computation resources for the traffic signal detection module is further broken down in Table 4.2.**

| | **% Application CPU** | **% System CPU** | **CPU Time (sec)** |
|---|---|---|---|
| **Traffic Signal Detection** | 66.57 | 51.12 | 1.02 |
| **Logging** | 21.94 | 16.85 | 0.34 |
| **Image Format Conversions** | 2.85 | 2.19 | 0.04 |
| **Communication** | 1.35 | 1.04 | 0.02 |
| **Schedule Prediction** | 0.52 | 0.40 | 0.01 |
| **Misc. (diplay, etc.)** | 6.77 | 5.20 | 0.10 |
| **Total** | 100.00 | 76.79 | 1.53 |

Table 4.2: **Computation resources for the different steps of SignalGuru's traffic signal detection algorithm.**

| | **% Application CPU** | **% System CPU** | **CPU Time (sec)** |
|---|---|---|---|
| **Color Filtering** | 7.71 | 5.92 | 0.12 |
| **Laplace Edge Detection** | 3.59 | 2.76 | 0.06 |
| **Hough Transform** | 30.55 | 23.46 | 0.47 |
| **Find max in Accumulator** | 21.64 | 16.62 | 0.33 |
| **Misc.** | 3.07 | 2.36 | 0.05 |
| **Total** | 66.57 | 51.12 | 1.02 |

signal schedule prediction takes less than $10ms$. For traffic-adaptive traffic signals, the prediction is based on the SVR models and takes $21ms$.

While in this configuration, SignalGuru was taking up only 77% of the system CPU time, it was not practically possible to further increase the video frame capture and traffic signal detection frequency. We found that when increasing this frequency by 20%, the GPS location updates would become very infrequent ($<0.1Hz$), as a result of the too high computational load. Such a low GPS location update frequency had detrimental effects, as SignalGuru's traffic signal detection was not activated/deactivated promptly on approaching/leaving a signalized intersection.

The average memory footprint of SignalGuru was 120.0 MB. However, only 20.1 MB, on average, were kept in actual RAM. The remaining 99.9 MB were assigned by the iOS to virtual memory. The iPhone 3GS devices have 256 MB of eDRAM. SignalGuru thus takes up only $\sim$8% of the device's actual RAM resources.

While the communication and memory overheads of SignalGuru are limited, SignalGuru's traffic signal detection is a major CPU hog. The video frame capture and the traffic signal detection frequency could be scaled down to reduce the computation resources of SignalGuru. However, it is critical to keep it as high as possible so that the traffic signal schedule prediction error is kept minimal. Therefore, SignalGuru's traffic signal detection module should be used judiciously. This is achieved in three ways:

1. First, the SignalGuru service is spawned on *only as many nodes* as necessary. The RegReS middleware takes care of that keeping the aggregate computation resources across a region at bay.
2. Second, the traffic signal detection module is activated on a carrier *only when* necessary. A SignalGuru carrier will try to detect traffic signals only when it is close enough to an intersection and thus a traffic signal could potentially be in sight. As explained earlier, the automatic activation/deactivation of the traffic signal detection module is based on the GPS location of the device and its distance from the traffic signal.
3. Third, *only the potentially useful part of an image* is processed for the purpose of detecting traffic signals. This is achieved with the IMU-based detection scheme. Thanks to this scheme, the traffic signal detection time is halved.

### 4.7.6 GLOSA Fuel Efficiency

For evaluating GLOSA, we used a 2.4L Chrysler PT Cruiser '01 city vehicle. We measured its fuel efficiency by connecting to its Controller Area Network (CAN) with a Scan Tool OBD-LINK device. The fuel efficiency was calculated based on the Intake Manifold Absolute Pressure (IMAP) approach with the help of the OBDwiz software.

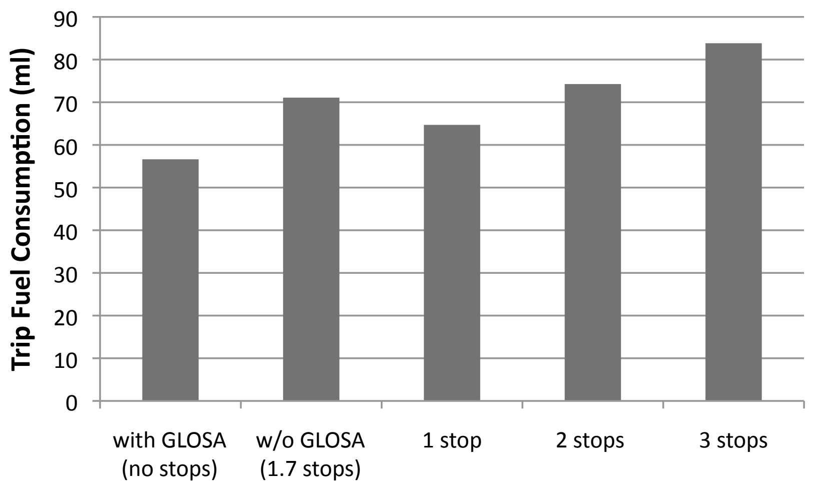


Figure 4.23: **GLOSA fuel efficiency evaluation.**

The trip starts at $P_1$ and ends at $P_2$ as shown in Figure 4.9, including the three intersections in our Cambridge deployment.

The driver completed 20 trips, following GLOSA's accurate advisory ($< 1s$ mean absolute prediction error[12]) at odd-numbered trips, and driving normally (without GLOSA's advisory) at even-numbered trips. When following GLOSA's advisory, the driver was able to avoid most stops (the driver sometimes had to brake because of pedestrians or cars in front). When not, the driver had to stop from zero to three times during each of the trips. As shown in Figure 4.23, GLOSA can offer significant savings reducing fuel consumption, on average, by 20.3% (from $71.1ml$ to $56.6ml$). In other words, GLOSA improves the vehicle's mileage, on average, by 24.5% (from $16.1mpg$ to $20.1mpg$).

## 4.8 Complex Intersections

In this section, we discuss practical issues regarding the operation of SignalGuru in complex intersections, as well as how SignalGuru can overcome them. In a complex

[12]For small distances from traffic signals, we found that it is more beneficial to provide the driver with the transition time instead of the recommended speed. First, for small distances ($<50m$) the GPS error is significant and second, the driver can better account for vehicles stopped at the intersection and time their acceleration appropriately.

intersection with many traffic signals, SignalGuru must be able to detect the correct traffic signal and also identify to which vehicular movement the traffic signal being detected corresponds.

### 4.8.1 Traffic Signal Detection

In a complex intersection with many traffic signals, SignalGuru will normally still detect the correct traffic signal, *i.e.,* the one that corresponds to the road segment that the vehicle is currently on. Normally, a vehicle that is approaching an intersection on a given road segment will be able to view only the corresponding traffic signal at a zero-degree angle. The traffic signals of the other road segments may be still within the camera's field of view, but will be seen at some angle. At angles $> 90^\circ$ the traffic signal bulbs will not be visible. At smaller angles, the bulbs will be visible, but will recorded on the video frame as ellipses instead of circles. Furthermore, these ellipses will be partially occluded by the traffic signal housing visors. While SignalGuru can still detect partially deformed and occluded traffic signals, its Hough transform voting algorithm will favor the most round and less occluded traffic signal, *i.e.,* the one that corresponds to the road segment that the vehicle is currently on.

Moreover, information about the exact location of traffic signals at an intersection can be leveraged to further narrow down the size of the IMU-based detection window. In this way, both the accuracy and the speed of the traffic signal detection will get improved. The locations of the traffic signals can be detected and recorded by the SignalGuru devices themselves.

### 4.8.2 Traffic Signal Identification

In a complex intersection with more than one traffic signal, SignalGuru needs to identify which specific traffic signal it is detecting, *i.e.,* to which direction of movement the detected traffic signal corresponds. While GPS localization can be used to identify the intersection, it is often not accurate enough to distinguish between the different road segments of an intersection. The identification of the traffic signal being detected is necessary in order to appropriately merge the data detected across different vehicles.

The traffic signal is identified based on the GPS heading (direction of movement) of the vehicle that is detecting it, as well as the shape of the traffic signal (round, right/left turn arrow, *etc.*). The heading of the vehicle is used to identify the road segment on which the vehicle is located by matching its reported GPS heading ($d^\circ + \delta^\circ$, where $\delta^\circ$ is the measurement error) to road segment that has the closest orientation ($d^\circ$). The number and orientation of the intersecting road segments at any given intersection can either be acquired by mapping information [110] or learnt by clustering movement traces of SignalGuru-enabled vehicles. Then, for example, a vehicle can tell that the signal it is detecting is for vehicles that want to turn left and are approaching the intersection on the road segment that is attached to the intersection at $d^\circ$ degrees compared to the geographic north.

## 4.9 Other Possible SignalGuru-enabled Applications

Besides GLOSA, a SignalGuru service that brings traffic signal schedule information to a driver's phone can enable several additional novel applications. These applications can help drivers reduce fuel consumption, reduce environmental impact, reduce travel time and increase safety.

Each of these applications comes with different requirements in terms of traffic signal schedule prediction accuracy and how much time in advance the predictions *should* be available. We term the latter *critical lead-up time*. How many seconds in advance the prediction is actually made available is termed *available lead-up time*. The vehicle's distance from the intersection stop line corresponding to the *critical lead-up time* when the vehicle is moving at the maximum allowed speed is termed

Table 4.3: **Application requirements on SignalGuru's traffic signal predictions.**

| | critical lead-up time (sec) | critical distance (m) | accuracy (sec) |
|---|---|---|---|
| Green Light Optimal Speed Advisory | 20 | 270 | <5 |
| Traffic Signal-Adaptive Navigation | 115 | 1500 | <9 |
| Red Light Duration Advisory | $\geq 5$ | 0 | <9 |
| Imminent Red Light Advisory | 20 | 300 | <5 |
| Red Light Violation Advisory (dry pavement) | 1.7 | 23 | <5 |
| Red Light Violation Advisory (wet pavement) | 2.7 | 37 | <5 |

*critical distance.* Table 4.3 shows the critical lead-up time and distance calculated for different applications, each assuming a traffic signal phase length (green light duration) of $47s$[13], and a speed limit of $30mph \approx 50km/h$.

Potential applications that can be enabled by SignalGuru's traffic signal schedule predictions include:

**Traffic Signal-Adaptive Navigation:** The travel time for a given trip can be reduced by advising drivers on possible detours that will let them avoid long waits at red lights. The average waiting time at a traffic signal is several tens of seconds and can be up to a few minutes [79]. A TSAN application, based on the traffic signal schedule that SignalGuru predicted, can calculate the travel time savings of possible detours and make suggestions to the driver accordingly. The critical lead-up time depends on the structure of the road network. However, predictions that are available while a vehicle is still five blocks away from a traffic signal ($1500m$ or $115s$ at $50km/h$) should provide enough headway for efficient detours for most road networks. A prediction error that is less than 20% of a traffic signal's phase length is desired in order to avoid suggesting unnecessary detours.

**Red Light Duration Advisory:** If the GLOSA or the TSAN applications cannot provide efficient suggestions to the driver, then the driver will have to wait at the

[13]Average phase length of traffic signals in Bugis, *i.e.,* Singapore's downtown (Section 4.6.2).

traffic signal. In this case, the Red Light Duration Advisory application can advise the driver on the residual amount of time before the light turns green, in other words, the amount of time the driver will have to wait. Drivers may then choose to switch off a vehicle's engine to save gas and decrease environmental impact. Restarting one's engine takes the same amount of fuel as idling for only five seconds [4], so the prediction critical lead-up time should be at least five seconds to yield benefits. The prediction accuracy should be significantly better ($<$20%) than the average red light waiting time so that drivers are not falsely advised to switch off their engines.

**Imminent Red Light Advisory:** The Imminent Red Light Advisory application advises the driver about the residual amount of time before the traffic signal ahead turns red. This application raises safety concerns, as the drivers may be tempted to exceed the speed limit in order to cross the intersection while the traffic signal is still green. Requirements are the same as for GLOSA.

**Red Light Violation Advisory:** The Red Light Violation Advisory application warns drivers when they are about to violate a red light. We can use the filtering and deglitching, as described in the previous sections, to validate and de-noise the red lights as detected by the camera of the on-board phone. When the signal ahead is red and the phone's accelerometer indicates the car is not decelerating, this application warns the driver, in order to prevent accidents and traffic tickets. The critical lead-up time can be as low as a few seconds just to allow the driver enough time to brake before entering the intersection. The critical distance corresponds to a vehicle's braking distance, when the vehicle is traveling at $30mph$ (speed limit).

## 4.10 Other Possible Camera-based Services

Besides SignalGuru, windshield-mounted smartphones can support a rich set of novel collaborative services with their cameras. We briefly describe here four additional services:

**Parking Space Availability Service:** Significant amount of fuel and time resources are wasted in the search of a parking spot in busy city centers [98]. At the same time, the circling of blocks in search of a parking spot further aggravates congestion problems. Windshield-mounted smartphone cameras can process captured images to discover available parking spots and subsequently disseminate information about their location to drivers that are looking for a free spot. As shown in Figure 4.24, free parking spots can be discovered by detecting features like the wheels of parked cars and the lines that segment parking spots. Such a system could be used as a stand-alone service or to augment proposed parking spot discovery systems that are based on ultrasonic rangefinders [98] or parking spot release reports (Section 3.5.4).

**Bus Localization and Arrival Time Service:** Services that inform the passengers about the time that their bus is expected to arrive have become very popular. However, such services are available in only a few cities as they necessitate the installation of specialized equipment in the buses.

Windshield-mounted smartphones could be used to support a grassroots bus localization and arrival time service in cities that do not have the necessary infrastructure. Windshield-mounted smartphone cameras can detect and identify buses by detecting their license plates and IDs. Bus IDs are often not only printed on both sides (front, back) of a bus, but also displayed in LED signs. Bright LED signs make the bus detection and identification task significantly easier. On encountering and identifying a bus, smartphones can estimate and disseminate the bus arrival time based on the their location and the prevailing traffic conditions [145].

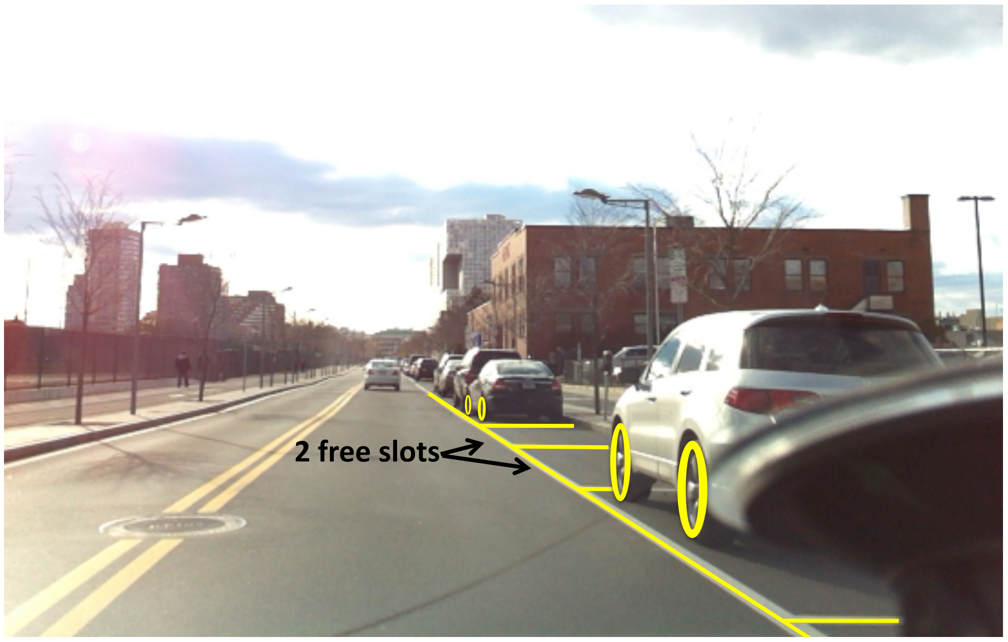


Figure 4.24: **Parking service enabled by windshield-mounted smartphones and their cameras. The detected vehicle wheels and parking spot lines are shown with yellow ovals and lines, respectively. Two free spots are detected.**

**Taxi Discovery Service:** The search for a taxi is a stressful and often time-consuming task. In some cities (*e.g.,* Singapore), taxis have LED signs on their roof that show whether the taxi is free ("TAXI" shown in green) or busy ("BUSY" shown in red). Windshield-mounted smartphones could detect the color-coded text displays to discover free taxis and inform prospective passengers about where they can find one.

**Cheap Gasoline Advisory Service:** Windshield-mounted smartphone cameras could detect the large signs that typically gasoline stations have and read off the gasoline prices. The gasoline prices would then be disseminated and shared with other vehicles to collaboratively support a service that helps drivers find the cheapest gasoline around them.

While detecting available parking spots or reading off gasoline prices from a gasoline station sign may be harder computationally than detecting a traffic signal, the time constraints are not as tight. Traffic signal detection needs to be fast so that new frames can be processed as frequently as possible and the traffic signal transition times are detected as accurately as possible. Particularly for the cheap gas advisory service, the time constraints are very loose. A windshield-mounted smartphone could capture an image and take several seconds or even minutes to process it and detect the prices. To improve the detection accuracy, still images could be captured instead of the video frames that we used to increase detection speed for SignalGuru.

## 4.11 Related Work

Several collaborative systems have been proposed that crowd-source information from the GPS, accelerometer and proximity sensors of everyday users in order to estimate traffic conditions [55, 99, 145], detect road abnormalities [33], collect information for available parking spots [98], compute fuel-efficient routes [44] and provide taxi ride fare and duration predictions [7]. In [89], Lee *et al.* propose an application that lets police track the movement of suspicious vehicles based on information sensed by camera-equipped vehicles. Other works have also proposed to equip vehicles with specialized cameras and detect traffic signals with the ultimate goal of enabling autonomous driving [47], assisting the driver [109], or detecting the location of intersections and overlaying navigation information [140]. In [26, 151], the authors enforce traffic laws (*e.g.,* detection of red light runners) by detecting traffic signals and their current status with stationary cameras affixed to infrastructure. Furthermore, as we discussed in the introduction, approaches aimed at enabling GLOSA have been based on costly infrastructure and, hence, failed to grow in scale. To the best of our knowledge, no other work has proposed to leverage commodity windshield-mount smartphone cameras, or above all, to *predict* the *future* schedule of traffic signals for the purpose of providing it to users and enabling the proposed set of novel applications.

Our camera-based traffic signal detection algorithm draws from several schemes mentioned above [47, 109, 140]. However, in contrast to these approaches that detect a single target, SignalGuru uses an iterative threshold-based approach for identifying valid traffic signal candidates. We also propose the IMU-based detection window scheme that leverages information from smartphones' accelerometer and gyro devices to narrow down the detection area, offering significant performance improvements. In order to be able to detect ill-captured traffic signals under poor ambient light conditions, previous approaches either use normalized RGB images [109] or estimate ambient illumination [26]. In contrast to these approaches, we leverage the observation that LED traffic signals are a light source of a fixed luminous intensity, and provide mechanisms to perform a one-time automatic adjustment of the smartphone camera's exposure setting. In this way, the camera hardware is configured to capture traffic signal bulbs correctly, regardless of the prevailing ambient light conditions, obviating the need for additional image processing steps. Last and most important, all these prior works focus solely on reporting the current status of traffic signals. They are not concerned with phase transitions and thus do not propose schemes to filter them, as they are not trying to collate the past traffic signal schedule for prediction of the future.

## 4.12 Summary

This chapter demonstrated the potential of RegReS-enabled platforms of collaborating mobile devices to fully support geo-locality services, without cloud servers and the associated long-range Internet communications. More specifically, this chapter presented novel collaborative services that are based on windshield-mounted smartphones and their cameras while focusing on SignalGuru, a traffic signal schedule advisory service. SignalGuru leverages the cameras of windshield-mounted smartphones to collaboratively detect and predict the schedule of traffic signals.

The case of SignalGuru highlighted the importance of collaboration, adaptation and the exploitation of smartphones' rich capabilities. By leveraging the inertial sensors of smartphones to selectively process parts of the captured images and introducing lightweight filtering layers, we showed that near real-time and accurate image based detection is possible on commodity smartphones. Furthermore, thanks to collaboration and adaptive Support Vector Regression models, SignalGuru could accurately predict not only pre-timed but also state-of-the-art traffic adaptive traffic signals. On average, SignalGuru comes within $0.66s$ for pre-timed traffic signals and within $2.45s$ for traffic-adaptive traffic signals. Particularly for the latter, without collaboration and adaptation, the prediction error would be $4.5\times$ higher.

Many more geo-locality services and associated applications can be supported by RegReS-enabled platforms delivering significant benefits. Besides the SignalGuru-enabled GLOSA that helps drivers save, on average, 20% on fuel, SignalGuru's traffic signal predictions enable more applications that can help drivers save fuel, reduce their environmental impact, reduce travel time and increase safety. Furthermore, windshield-mounted smartphones with their cameras can capture content-rich information of the road ahead, and besides traffic signals, detect parking spot availabilities, cheap gasoline stations, free taxis and the current location of city buses.

Overall, this chapter demonstrated the great potential of RegReS-enabled confederations of collaborating mobile devices. As the case of SignalGuru demonstrated, such platforms are very powerful and can fully support even challenging camera-based services. In this way, by leveraging the power of mobile device confederations, the need for cloud servers and long-range communications to reach them is completely obviated for geo-locality services.

# Chapter 5

# Conclusions

This chapter begins by briefly reviewing the main contributions of the thesis in Section 5.1, before discussing some open questions and future research directions in Section 5.2.

## 5.1 Contributions Summary

This thesis explores challenges that arise in the design of emerging geo-locality and non-geo-locality mobile services. More specifically, it tackles the poor user experience problem of the existing mobile cloud service architecture by reducing its need for costly and slow long-range cellular communications. To reduce long-range cellular communications, this thesis proposes to leverage mobile device collaboration in order to enable efficient hosting of services on mobile devices themselves. While there has been a rich set of work on caching service data and supporting parts of service functionality on mobile devices for the purpose of reducing the frequency of long-range communications, the applications of such works have been constrained and often fail to judiciously exploit the capabilities and resources of mobile devices.

This thesis first analyzes mobile device technology trends and demonstrates which and how the ever increasing capabilities of mobile devices can be leveraged to effectively support, on mobile device themselves, services that have been traditionally thought to belong to the cloud. To achieve this goal, the thesis introduces two different service architectures for the two different types of mobile services: 1) traditional non-geo-locality services, and 2) emerging geo-locality services. To explore the proposed architectures, a service is extensively evaluated for each. At the same time, the work in this thesis highlights the importance of collaboration and adaptation in improving the performance of both these architectures and the services running atop. The lessons learned from our work will become even more significant as mobile devices become more and more powerful in terms of computational resources (CPU, memory), storage capacity, sensing capabilities and communication interfaces.

Overall the primary contributions of this thesis are the following:

1. **Non-geo-locality services:** The thesis analyzes NVM technology trends and demonstrates that several traditional mobile services that have so far resided only within the cloud can be effectively augmented by mobile device-resident caching architectures (pocket cloudlets) that are based on personal user and collaboratively-generated community access patterns. As our analysis shows, NVM may continue experiencing significant and steady improvements in density for at least ten more years. The abundance in memory capacity of mobile devices can be used to support pocket cloudlets for a rich set of mobile services, significantly reducing the need for costly long-range communications and thus greatly improving user experience. As shown in our evaluation for PocketSearch, a web search pocket cloudlet, the average user response time is reduced by 62% and power consumption by 63%. Furthermore, by serving requests locally from the pocket cloudlet, when possible, the use of costly and increasingly scarce long-range communications is reduced by 66%.

   The impact of this work is very substantial. Not only does this work significantly improve the performance of a specific popular mobile service (web search), but also provides, at the same time, a methodology and architecture to collaboratively and adaptively guide the selective replication of mobile services on mobile devices. The example of PocketSearch, in combination with promising NVM scaling trends, is expected to foster the augmentation of more mobile services with mobile device-resident caching architectures.

2. **Geo-locality services:** The thesis first studies the promising trends in computing and networking for mobile devices and then demonstrates that a rich set of geo-locality services can be fully supported on confederations of collaborating mobile devices completely obviating the need for slow, costly and potentially unavailable long-range communications. We identify the five major traits of geo-locality services when hosted on highly volatile and unreliable networks of everyday user mobile devices. Furthermore, we provide the first solution that accounts for all these traits by introducing the service carrier density metric. To maintain geo-locality services on such volatile and unreliable platforms, our work proposes and evaluates RegReS, a fully-distributed, collaborative and adaptive service carrier estimation and maintenance scheme. We show that with collaboration and adaptation, networks of mobile devices can accurately maintain a targeted service carrier density. The density mean absolute error remains below 16% across a wide range of configurations. Thanks to RegReS, the increased yet often constrained resources of mobile devices are judiciously used to support geo-locality services within a region and the need for long-range communications is completely obviated. In many cases, service access from local storage or over short-range communications takes significantly less than a second ($30ms$ for our Parking Availability Service) as opposed to several seconds ($> 5s$) over 3G.

   Besides introducing a new metric (service carrier density) and corresponding middleware for the collaborative maintenance of geo-locality services, this work introduces a more general collaborative and adaptive estimation scheme. We explore which dynamic factors influence the performance of the estimation scheme and propose a general model to adapt. At the same time, our work shows that static schemes fail very badly at tracking the existing density of service carriers. This raises concerns about the robustness and applicability of proposed approaches for challenging mobile networks. A collaborative and above all adaptive scheme, as introduced by RegReS, should be used in dynamic environments whether the ultimate goal is request routing [38, 42, 89], power management [156] or service replication like in RegReS. As shown in our results, non-collaborative and especially static schemes fail very badly at tracking the status of region-wide parameters, *e.g.,* service carrier density.

3. **Collaboration and adaptation:** Throughout the thesis, the importance of collaboration and adaptation is highlighted. Collaboration and adaptation significantly improve the performance of both the proposed mobile device-resident architectures and the services running atop. Collaboration almost doubles the carrier density estimation accuracy of RegReS. Furthermore, without adaptation, the carrier mean density error goes from 16% up to 41% for the cases tested, and could further increase arbitrarily in more extreme scenarios. The benefits are equally pronounced for the SignalGuru service as well. Thanks to collaboration and adaptation SignalGuru can accurately predict the schedule of both pre-timed (0.66$s$) and traffic-adaptive traffic signals (2.45$s$). Particularly for the more challenging traffic-adaptive traffic signals, collaboration and adaptation are crucial. Collaboration improves the prediction accuracy by 78% and adaptation by an additional 25%. For PocketSearch, collaboratively-generated community access patterns help improve the hit rate by 2% to 19% depending on the class of the user. Adaptive cache updates improve the hit rate by 2%. However, as discussed in Section 5.2.2, we believe that with more frequent updates and more

input signals to guide the decisions, the benefit of adaptive prefetching could become significantly higher.

4. **Novel grassroots services and benefits:** Besides the analysis of the promising mobile device technology trends, the thesis demonstrates with real-world deployments the potential of mobile devices to support a rich set of services without cloud servers and long-range communications to reach them. The thesis also demonstrates the benefits that such grassroots mobile device-resident services can deliver. SignalGuru, in particular, epitomizes this potential. As this service illustrates, even camera-based services that were thought to be prohibitively resource-intensive can be supported on confederations of collaborating mobile devices. Furthermore, SignalGuru's traffic signal schedule predictions can enable several applications to help drivers reduce their fuel consumption (20.3% for GLOSA), environmental impact and travel time. Last, the novel collaborative sensing platform that SignalGuru introduced can enable many more services for the discovery of free parking spots, cheap gasoline stations, free taxis and a commuter's current bus location. The societal and environmental benefits of such services are potentially very substantial.

## 5.2 Future Work

As mobile devices with short range communications will be becoming increasingly pervasive (*e.g.,* vehicles with DSRC communication) and long-range communication bandwidth will become an increasingly scarce resource, the need for device-resident services will become increasingly pressing. As shown in this thesis, in order to efficiently and effectively support mobile services on mobile devices, both collaboration and adaptation are critical. Here, we discuss avenues for future research on these two key aspects. We also discuss future work on collaborative services enabled by our proposed mobile service architectures.

### 5.2.1 Collaboration

As discussed and evaluated in Chapters 2, 3 and 4, collaboration is of vital importance. The Pocket Cloudlets architecture depends on mobile users to contribute and share their personal user models so that community models can be constructed and, subsequently, cloud service content is selectively cached, ranked and prefetched. RegReS relies on confederations of mobile devices to collaborative maintain services across time within a defined region. Opportunistic collaboration is also leveraged to improve the accuracy of density estimations. Furthermore, collaboration is critical for SignalGuru in order to enable advance and accurate traffic signal schedule advisory.

When enabling such collaborative ecosystems, several challenges emerge that need further research:

**Incentive Mechanisms and User Policies:** Proper incentives need to be provided to mobile users so that they participate in collaborative service provision. Proposed price- and budget-based schemes [1, 51, 150] that enable multilateral exchanges can be used as the foundation.

However, incentivizing mobile users will typically be harder and more challenging. Mobile environments, as opposed to desktop environments, are resource-constrained. Battery power, compute resources and particularly long-range communications are limited and often scarce. Every extra packet of communication may also cost extra and increase the monthly bill of the user. Mobile users will typically have more to sacrifice when taking part in the collaborative provision of a service. At the same time, the need emerges to provide the proper tools and interfaces to let users customize their participation policies and set their own "price" for contributing their device to the collaborative ecosystem.

In RegReS, we favored hosting the service on nodes that will be around longer (with longer estimated residual residence time). Alternatively, collaborative middleware, like RegReS, could favor nodes that have low credits, *i.e.,* nodes that have barely participated so far. Alternatively, in a service provision market where services pay credits to mobile devices that participate in the service maintenance, services may opt for devices that have a better service provision reputation or charge the minimum amount of credits for a given service provision quality.

Furthermore, a scheme is needed to control the amount of regional resources (target carrier density) that services ask from RegReS. Some regional authority could regulate this. A market-based system in which services have to pay mobile devices for their hosting could also be an alternative approach. The viability of such approaches needs further research.

In summary, schemes are necessary to control which entities and to what extent are allowed to inject collaborative services, how mobile devices will be rated and selected to host a given service and how mobile users will be incentivized to contribute their device's resources.

**Security, Privacy, Trust:** The potential incentives and resulting attacks of adversarial and malicious users need to be analyzed on a per service and per application basis. Based on such analyses, proposed schemes for service security and trust establishment [90, 131] need to be revisited. At the same time, the more a node collaborates, the more the node gets exposed and potentially sacrifices its privacy. Privacy, security and trust should all be safeguarded in order to encourage collaboration. This can be best achieved on a customized per service and per application basis.

**Power Management:** Mobile devices are typically battery-powered and operate on constrained energy budgets. Although RegReS's collaboration scheme is lightweight and trades off inefficient long-range communications for short-range communications, it is not clear that such a mobile service architecture can lead to energy savings; short-range communications may be significantly more power-efficient per bit sent or received, but commercial long-range communications (*e.g.,* 3G) are already employing aggressive power-saving mechanisms. These mechanisms put the communication interface to sleep when it is not being used. When in sleep mode, long-range communication interfaces spend far less power than short-range communication interfaces that are idle as well, but awake (powered-on but not sending or receiving data). Proposed approaches for sleep schedule-based energy conservation and lifetime maximization [20, 75, 156] in sensor networks should be adjusted and integrated in RegReS's collaborative density maintenance scheme. While the resulting energy savings depend a lot on the properties of the service and the underlying mobile device composition, when coupled with such schemes, short-range communications have great potential of significantly reducing energy consumption over their long-range counterparts.

### 5.2.2 Adaptation

Adaptation is critical in highly dynamic environments whether the goal is to track service carrier density, track search trends or predict the behavior of systems that adapt to dynamic conditions, *e.g.,* traffic-adaptive traffic signals that adapt to vehicle flows. The lessons learnt about the importance of adaptation, in general, and adaptive estimation schemes like the one that this thesis proposed, in particular, should be employed, when using distributed estimation to perform request-routing decisions [38, 42, 89], power management [156] or service replication (RegReS). As our results showed, static estimation-based schemes are highly likely to break when key system parameters (*e.g.,* node mobility, *etc.*) change. Proposed estimation-based schemes for mobile networks should be re-tested under dynamic conditions and made adaptive.

Furthermore, in order to increase the ability of the proposed systems to adapt, more information could be leveraged. For example, in the case of Pocket Cloudlets, social network (facebook, twitter, foursquare, *etc.*) data streams (wall posts, tweets, *etc.*)

could be leveraged to detect emerging subjects within a user's network and prefetch the corresponding information (search results, web pages, maps, local business information, *etc.*) on the user's phone. Also, the prediction of a given intersection's traffic signal schedule could be improved by fusing information about the schedule of neighboring signalized intersections. Neighboring intersections are sometimes coordinated, *i.e.,* the schedules of their traffic signals are schedule highly correlated.

### 5.2.3 Services

The only non-geo-locality service that this thesis prototyped and evaluated is PocketSearch, a mobile search pocket cloudlet. The cacheability on the Pocket Cloudlet architecture of the services described in Section 2.2 can be analyzed more thoroughly and the performance benefits of their implementation as pocket cloudlets can be evaluated as well.

As regards geo-locality services, this thesis prototyped PAS, SignalGuru and GLOSA. The plausibility and efficiency of the other novel RegReS-enabled services envisioned in Chapter 4 will be interesting to explore as well. Furthermore, the impact of applications, like the SignalGuru-enabled GLOSA, can be tested on a larger scale by means of detailed microscopic simulation that models driver behavior and interaction with advisory systems.

This thesis explored geo-locality and non-geo-locality services. Hybrid services and corresponding architectures will be interesting to explore as well. Some services may have a strong geo-locality component (data generated within a specific region and many users accessing within the very same region), but at the same time also have a potentially less strong yet still significant non-geo-locality component (data often accessed by users that are farther away) as well. For example, drivers are typically interested in finding a free parking spot in their vicinity (at most a couple blocks away). These free parking spots are discovered by other vehicles traveling through the same region. This means that a parking service is a good example of a geo-locality service. However, some drivers may also be interested in learning about aggregate parking statistics before they drive to a specific area so that, for example, they can determine if they would be able to easily find a free parking spot and it would thus be a good idea to use their car, instead of some other means of transportation. A solution would be to maintain two disjoint and independent parking services, one as a pocket cloudlet and the other one as a RegReS service. A hybrid architecture, however, that enables these two related services to interact could potentially constitute a significantly more effective solution.

## 5.3 Summary

This thesis explores challenges pertaining to the design of emerging geo-locality and non-geo-locality mobile services. More specifically, this thesis proposes novel mobile service architectures that help alleviate the need for costly, slow and scarce long-range cellular communications, thus, significantly improving the poor mobile user experience. The Pocket Cloudlets architecture greatly reduces the long-range communication bandwidth for traditional non-geo-locality services by selectively caching parts of them on mobile devices. For emerging geo-locality-services, the proposed RegReS middleware hosts the services fully on mobile devices, completely alleviating the need for long-range communications. By prototyping and evaluating these novel mobile service architectures and services running atop them, this thesis demonstrates that services can be effectively hosted on mobile devices and enable a rich set of novel applications. Furthermore, this thesis illustrates that the challenges of dynamic mobile environments can be effectively tackled by means of collaboration and adaptation. I believe that this work will motivate further research in collaborative mobile device-resident services, in general, and camera-based services, in particular.

# Bibliography

[1] Christina Aperjis, Michael J. Freedman, and Ramesh Johari. Peer-assisted Content Distribution with Prices. In *Proceedings of the ACM CoNEXT Conference*, 2008.

[2] ARM Cortex-A Comparison. `http://www.arm.com/products/processors/cortex-a/index.php`.

[3] ARM11MPCore Processor. `http://www.arm.com/products/processors/classic/arm11/arm11-mpcore.php/`.

[4] American Society of Mechanical Engineers (ASME) - Florida Section. `http://memagazine.asme.org/articles/2007/june/Idle\_vs\_Restart.cfm`.

[5] Audi Travolution Project. `http://www.audi.com/com/brand/en/tools/news/pool/2010/06/audi_travolution_.html`.

[6] Maurice. J. Bach, Mark. W. Luppi, A. S. Melamed, and Kang Yueh. A Remote-file Cache for RFS. In *Proceedings of the USENIX*, 1987.

[7] Rajesh Krishna Balan, Khoa Xuan Nguyen, and Lingxiao Jiang. Real-time Trip Information Service for a Large Taxi Fleet. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2011.

[8] Luiz André Barroso. The Price of Performance. *ACM Queue*, 2005.

[9] Luiz André Barroso and Urs Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.

[10] BeiDou (COMPASS) Global Navigation Satellite System. `http://www.beidou.gov.cn/`.

[11] Laszlo A. Belady. A Study of Replacement Algorithms for a Virtual-Storage Computer. *IBM Systems Journal*, 1966.

[12] Paolo Bellavista, Antonio Corradi, and Eugenio Magistretti. REDMAN: A Decentralized Middleware Solution for Cooperative Replication in Dense MANETs. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW)*, 2005.

[13] Edward Benson, Adam Marcus, David Karger, and Samuel Madden. Sync Kit: a Persistent Client-side Database Caching Toolkit for Data Intensive Websites. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2010.

[14] Azer Bestavros. Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 1996.

[15] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-level Document Caching in the Internet. In *Proceedings of the International Workshop on Services in Distributed and Networked Environments (SDNE)*, 1995.

[16] Nabhendra Bisnik, Alhussein Abouzeid, and Volkan Isler. Stochastic Event Capture using Mobile Sensors subject to a Quality Metric. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2006.

[17] Specification of the Bluetooth System – Wireless Connections Made Easy – Specification Volumes 0-5. `https://www.bluetooth.org/Technical/Specifications/adopted.html/`.

[18] Peter J. Braam. The CODA Distributed File System. *Linux Journal*, 1998.

[19] Murat Caliskan, Daniel Graupner, and Martin Mauve. Decentralized Discovery of Free Parking Places. In *Proceedings of the International Workshop on Vehicular Inter-Networking (VANET)*, 2006.

[20] Qing Cao, Tarek Abdelzaher, Tian He, and John Stankovic. Towards Optimal Sleep Scheduling in Sensor Networks for Rare-event Detection. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[21] Car 2 Car Communication Consortium Manifesto. `http://www.car-2-car.org/index.php?id=31`, 2007.

[22] Peter B. Catrysse and Brian A. Wandell. Roadmap for CMOS Image Sensors: Moore Meets Planck and Sommerfeld. In *Proceedings of the SPIE*, 2005.

[23] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2005.

[25] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor Location Sensing Using Geo-magnetism. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2011.

[26] Yun-Chung Chung, Jung-Ming Wang, and Sei-Wang Chen. Vision-based Traffic Light Detection System at Intersections. *Journal of Taiwan Normal University: Mathematics, Science and Technology*, 2002.

[27] Karen Church, Barry Smyth, Keith Bradley, and Paul Cotter. A Large Scale Study of European Mobile Search Behavior. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, 2008.

[28] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. Mobile Information Access: A Study of Emerging Search Behavior on the Mobile Internet. *ACM Transactions on the Web*, 2007.

[29] Capacity Coverage & Deployment Considerations for IEEE 802.11g, CISCO White Paper. `http://www.cisco.com/en/US/products/hw/wireless/ps4570/products_white_paper09186a00801d61a3.shtml`, 2009.

[30] Gurcan Comert and Mecit Cetin. Queue Length Estimation from Probe Vehicle Location and the Impacts of Sample Size. *European Journal of Operational Research*, 2009.

[31] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards Mobile Phone Localization without War-driving. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2010.

[32] The Evolution of EDGE, Ericsson White Paper. `http://www.ericsson.com/res/docs/whitepapers/evolution_to_edge.pdf`, 2009.

[33] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2008.

[34] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark Silicon and the End of Multicore Scaling. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2011.

[35] Facebook. `http://www.facebook.com/`.

[36] Tiziano Fagni, Raffaele Perego, Fabrizio Silvestri, Salvatore Orlando, University Ca, and Foscari Venezia. Boosting the Performance of Web Search Engines: Caching and Prefetching Query Results by Exploiting Historical Usage Data. *ACM Transactions on Information Systems*, 24, 2006.

[37] Gregory Finn. Routing and Addressing Problems in Large Metropolitan-Scale Internetworks. Technical Report Research Report ISI/RR-87-180, Information Sciences Institute, University of Southern California, 1987.

[38] Marco Fiore, Claudio Casetti, and Carla-Fabiana Chiasserini. Efficient Retrieval of User Contents in MANETs. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2007.

[39] Wayne Fisher. IEEE 802.11p Wireless LANs: Development of DSRC/WAVE Standards. Technical Report IEEE 802.11-07/2045r0, ARINC, Inc, 2007.

[40] International Technology Roadmap for Semiconductors Working Group. International Technology Roadmap for Semiconductors. Technical report, 2009.

[41] Michael J. Franklin, Michael J. Carey, and Miron Livny. Local Disk Caching for Client-Server Database Systems. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1993.

[42] Roy Friedman and Noam Mori. 3DLS: Density-driven Data Location Service for Mobile Ad-hoc Networks. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2009.

[43] Galileo Global Navigation Satellite System. `http://www.esa.int/esaNA/galileo.html/`.

[44] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. GreenGPS: A Participatory Sensing Fuel-efficient Maps Application. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2010.

[45] Raghu K. Ganti, Nam Pham, Yu-En Tsai, and Tarek F. Abdelzaher. PoolView: Stream Privacy for Grassroots Participatory Sensing. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[46] GLONASS Global Navigation Satellite System. `http://www.glonass-ianc.rsa.ru/en/index.php/`.

[47] Jianwei Gong, Yanhua Jiang, Guangming Xiong, Chaohua Guan, Gang Tao, and Huiyan Chen. The Recognition and Tracking of Traffic Lights Based on Color Segmentation and CAMSHIFT for Intelligent Vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2010.

[48] James R. Goodman. Using Cache Memory to Reduce Processor-Memory Traffic. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1983.

[49] Marco Gruteser and Dirk Grunwald. Anonymous Usage of Location-based Services through Spatial and Temporal Cloaking. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2003.

[50] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving Ads from Localhost for Performance, Privacy, and Profit. In *Proceedings of the Workshop on Hot Topics in Networks (HotNets)*, 2009.

[51] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A Reputation System for Peer-to-Peer Networks. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2003.

[52] James Gwertzman and Margo Seltzer. The Case for Geographical Push-caching. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, 1995.

[53] Pradip Hari, Kevin Ko, Emmanouil Koukoumidis, Ulrich Kremer, Margaret Martonosi, Desiree Ottoni, Li-Shiuan Peh, and Pei Zhang. SARANA: Language, Compiler, and Runtime System Support for Spatially-Aware and Resource-Aware Mobile Computing. *Philosophical Transactions of the Royal Society*, 2008.

[54] Pradip Hari, John McCabe, Jonathan Banafato, Marcus Henry, Kevin Ko, Emmanouil Koukoumidis, Ulrich Kremer, Margaret Martonosi, and Li-Shiuan Peh. Adaptive Spatiotemporal Node Selection in Dynamic Networks. In *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques (PACT)*, 2010.

[55] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan-Carlos Herrera, Alexandre M. Bayen, Murali Annavaram, and Quinn Jacobson. Virtual Trip Lines for Distributed Privacy-Preserving Traffic Monitoring. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2008.

[56] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Enhancing Security and Privacy in Traffic-Monitoring Systems. *IEEE Pervasive Computing*, 2006.

[57] Yiming Huai. Spin-transfer Torque MRAM (STT-MRAM): Challenges and Prospects. *Association of Asia Pacific Physical Societies (AAPPS) Bulletin*, 2008.

[58] Junxian Huang, Qiang Xu, Birjodh Tiwana, Z. Morley Mao, Ming Zhang, and Paramvir Bahl. Anatomizing Application Performance Differences on Smartphones. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2010.

[59] IEEE 802.11-2007. IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report IEEE Std 802.11-2007, IEEE, 2007.

[60] IEEE 802.11n-2009. IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 5: Enhancements for Higher Throughput. Technical Report IEEE Std 802.11n-2009, IEEE, 2009.

[61] IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 6: Wireless Access in Vehicular Environments. Technical Report IEEE Std 802.11p-2010, IEEE, 2010.

[62] IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements. Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). Technical Report IEEE Std 802.15.1-2005, IEEE, 2005.

[63] Framework and Overall Objectives of the Future Development of IMT 2000 and Systems beyond IMT 2000. Technical Report Report ITU-R M.1645, International Telecommunications Union, 2003.

[64] Requirements Related to Technical Performance for IMT-Advanced Radio Interface(s). Technical Report Report ITU-R M.2134, International Telecommunications Union, 2008.

[65] Sibren Isaacman and Margaret Martonosi. The C-LINK System for Collaborative Web Usage: A Real-World Deployment in Rural Nicaragua. In *Proceedings of the ACM Workshop on Networked Systems for Developing Regions (NSDR)*, 2009.

[66] Jorjeta G. Jetcheva, Yih-Chun Hu, Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. CRAWDAD Data Set Rice/Ad-Hoc-City (v. 2003-09-11).

[67] Colin Johnson. Memristors Ready for Prime Time. `http://www.eetimes.com/electronics-news/4077811/Memristors-ready-for-prime-time`, 2008.

[68] Maryam Kamvar and Shumeet Baluja. A Large Scale Study of Wireless Search Behavior: Google Mobile Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2006.

[69] Maryam Kamvar and Shumeet Baluja. Deciphering Trends in Mobile Search. *Computer Journal, IEEE Computer Society Press*, 2007.

[70] Maryam Kamvar and Shumeet Baluja. The Role of Context in Query Input: Using Contextual Signals to Complete Queries on Mobile Devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, 2007.

[71] Maryam Kamvar and Shumeet Baluja. Query Suggestions for Mobile Search: Understanding Usage Patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2008.

[72] Maryam Kamvar, Melanie Kellar, Rajan Patel, and Ya Xu. Computers and iPhones and Mobile Phones, oh my!: A Logs-based Comparison of Search Users on Different Devices. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2009.

[73] Elliott D. Kaplan. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[74] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[75] Gaurav S. Kasbekar, Yigal Bejerano, and Saswati Sarkar. Lifetime and Coverage Guarantees through Distributed Coordinate-free Sensor Activation. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2009.

[76] DerChang Kau, Stephen Tang, Ilya V. Karpov, Rick Dodge, Brett Klehn, Johannes A. Kalb, Jonathan Strand, Aleshandre Diaz, Nelson Leung, Jack Wu, Sean Lee, Tim Langtry, Kuo wei Chang, Christina Papagianni, Jinwook Lee, Jeremy Hirst, Swetha Erra, Eddie Flores, Nick Righos, Herman Castro, and Gianpaolo Spadini. A Stackable Cross Point Phase Change Memory. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, 2009.

[77] Yoonheui Kim, Victor Lesser, Deepak Ganesan, and Ramesh Sitaraman. Cluster-Swap: A Distributed k-median Algorithm for Sensor Networks. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2009.

[78] Jonathan G. Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. Implications of Historical Trends in the Electrical Efficiency of Computing. *IEEE Annals of the History of Computing*, 2011.

[79] Peter Koonce, Lee Rodegerdts, Lee Kevin, Shaun Quayle, Scott Beaird, Cade Braud, Jim Bonneson, Phil Tarnoff, and Tom Urbanik. *Traffic Signal Timing Manual*. U.S. Department of Transportation, 2008.

[80] Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Jie Liu, and Doug Burger. Improving Mobile Search Experience with SONGO. *Microsoft Research Technical Report*, 2010.

[81] Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Karin Strauss, Jie Liu, and Doug Burger. Pocket Cloudlets. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011.

[82] Emmanouil Koukoumidis, Margaret Martonosi, and Li-Shiuan Peh. Leveraging Smartphone Cameras for Collaborative Road Advisories. *IEEE Transactions on Mobile Computing*, 2012.

[83] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Martonosi. Demo: SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys) Demo Session*, 2011.

[84] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Martonosi. RegReS: Adaptively Maintaining a Target Density of Regional Services in Opportunistic Vehicular Networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2011.

[85] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Martonosi. SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2011.

[86] Santosh Kumar, Ten H. Lai, and József Balogh. On k-coverage in a Mostly Sleeping Sensor Network. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[87] Nikolaos Laoutaris, Georgios Smaragdakis, Konstantinos Oikonomou, Ioannis Stavrakakis, and Azer Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2007.

[88] Kevin C. Lee, Seung-Hoon Lee, Ryan Cheung, Uichin Lee, and Mario Gerla. First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed. In *Proceedings of the IEEE Workshop on Mobile Networking for Vehicular Environments (MOVE)*, 2007.

[89] Uichin Lee, Eugenio Magistretti, Mario Gerla, Bellavista, and Antonio Corradi. Dissemination and Harvesting of Urban Data using Vehicular Sensor Platforms. *IEEE Transactions on Vehicular Technology*, 2008.

[90] Vincent Lenders, Emmanouil Koukoumidis, Pei Zhang, and Margaret Martonosi. Location-based Trust for Mobile User-generated Content: Applications, Challenges and Implementations. In *Proceedings of the International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2008.

[91] Ilias Leontiadis, Paolo Costa, and Cecilia Mascolo. Persistent Content-based Information Dissemination in Hybrid Vehicular Networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2009.

[92] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[93] Ting Liu and Margaret Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2003.

[94] Yajie Ma, Mark Richards, Moustafa Ghanem, Yike Guo, and John Hassard. Air Pollution Monitoring and Mining Based on Sensor Grid in London. *Sensors*, 2008.

[95] Christian Maihöfer, Tim Leinmüller, and Elmar Schoch. Abiding Geocast: Time-stable Geocast for Ad Hoc Networks. In *Proceedings of the International Workshop on Vehicular Inter-Networking (VANET)*, 2005.

[96] Evangelos Markatos. On Caching Search Engine Query Results. *Computer Communications*, 2000.

[97] Evangelos P. Markatos and Catherine E. Chronaki. A Top-10 Approach to Prefetching on the Web. In *Proceedings of INET*, 1998.

[98] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. ParkNet: Drive-by Sensing of Road-side Parking Statistics. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2010.

[99] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Using Mobile Smartphones for Rich Monitoring of Road and Traffic Conditions. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[100] Gordon E. Moore. Cramming more Components onto Integrated Circuits. *Electronics Magazine*, 1965.

[101] Mobile Internet Report, Morgan Stanley Research. `http://www.morganstanley.com/institutional/techresearch/mobile_internet_report122009.html`, 2009.

[102] Alok Nandan, Shirshanka Das, Giovanni Pau, Mario Gerla, and M. Y. Sanadidi. Co-operative Downloading in Vehicular Ad-hoc Wireless Networks. In *Proceedings of the International Conference on Wireless On-demand Network Systems and Services (WONS)*, 2005.

[103] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. A Data Mining Algorithm for Generalized Web Prefetching. *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[104] Julio C. Navas and Tomasz Imielinski. GeoCast - Geographic Addressing and Routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1997.

[105] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout. Caching in the Sprite Network File System. *ACM Transactions on Computer Systems*, 1988.

[106] Nokia Radar Sensor. `http://conversations.nokia.com/2010/01/27/nokia-mobile-radar/`.

[107] Kunle Olukotun and Lance Hammond. The Future of Microprocessors. *ACM Queue*, 2005.

[108] Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Ken Wilson, and Kunyung Chang. The Case for a Single-Chip Multiprocessor. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 1996.

[109] Masako Omachi and Shinichiro Omachi. Traffic Light Detection with Color and Edge Information. In *Proceedings of the IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, 2009.

[110] OpenStreetMap. `http://www.openstreetmap.org/`.

[111] Open-Access Research Testbed for Next-Generation Wireless Networks. `http://www.orbit-lab.org/`.

[112] Mockapetris P. RFC 1034: Domain Names - Concepts and Facilities. `http://tools.ietf.org/html/rfc1034`, 1987.

[113] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. *SIGCOMM Computer Communication Review*, 1996.

[114] Chunyi Peng, Guobin Shen, Yongguang Zhang, and Songwu Lu. Point&Connect: Intention-based Device Pairing for Mobile Phone Users. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2009.

[115] James Pitkow and Peter Pirolli. Mining Longest Repeating Subsequences to Predict World Wide Web Surfing. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1999.

[116] Rajib Kumar Rana, Chun Tung Chou, Salil S. Kanhere, Nirupama Bulusu, and Wen Hu. Ear-Phone: An End-to-End Participatory Urban Noise Mapping System. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2010.

[117] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change Random Access Memory: A Scalable Technology. *IBM Journal of Research and Development*, 2008.

[118] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramach, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2005.

[119] Muhammad Hassan Raza, Larry Hughes, and Imran Raza. *Trends in Intelligent Systems and Computer Engineering*, chapter Density: A Context Parameter of Ad Hoc Networks. Springer US, 2008.

[120] Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment Framework for Participatory Sensing Data Collections. In *Proceedings of the International Conference on Pervasive Computing (PERVASIVE)*, 2010.

[121] Peggy Reynolds, Julie Von Behren, Robert B Gunier, Debie E. Goldberg, Andrew Hertz, and Daniel Smith. *Cancer Causes Control*, chapter Traffic Patterns and Childhood Cancer Incidence Rates in California, United States. Springer Netherlands, 2002.

[122] Research and Innovative Technology Administration (RITA) | Intelligent Transportation Systems (ITS) | Deployment Statistics Database. `http://www.itsdeployment.its.dot.gov/Results.asp?rpt=M&filter=1&ID=320`.

[123] Research and Innovative Technology Administration (RITA) - U.S. Department of Transportation. `http://www.its.dot.gov/`.

[124] Ronny Ronen, Senior Member, Avi Mendelson, Konrad Lai, Shih lien Lu, Fred Pollack, John, and P. Shen. Coming Challenges in Microarchitecture and Architecture. In *Proceedings of the IEEE*, 2001.

[125] Peter Rysavy. EDGE, HSPA and LTE – Broadband Innovation, Rysavy Research White Paper. `http://www.rysavy.com/Articles/2008_09_Broadband_Innovation.pdf`, 2008.

[126] Peter Rysavy. HSPA to LTE-Advanced: 3GPP Broadband Evolution to IMT-Advanced (4G), Rysavy Research White Paper. `http://www.rysavy.com/Articles/2009_09_3G_Americas_RysavyResearch_HSPA-LTE_Advanced.pdf`, 2009.

[127] Peter Rysavy. Mobile Broadband Capacity Constraints and the Need for Optimization, Rysavy Research White Paper. `http://www.rysavy.com/Articles/2010_02_Rysavy_Mobile_Broadband_Capacity_Constraints.pdf`, 2010.

[128] Samsung ARM Application Processor. http://www.samsung.com/global/business/semiconductor/products/mobilesoc/Products_ApplicationProcessor.html/.

[129] Samsung's New 16-Megapixel CMOS Image Sensor Features High-Resolution Sensitivity and Performance for Advanced Mobile Applications. http://www.samsung.com/global/business/semiconductor/newsView.do?news_id=1262.

[130] T. Sanpechuda and L. Kovavisaruch. A Review of RFID Localization: Applications and Techniques. In *Proceedings of the International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2008.

[131] Stefan Saroiu and Alec Wolman. I am a sensor, and I approve this message. In *Proceedings of the International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2010.

[132] Sydney Coordinated Adaptive Traffic System (SCATS). http://www.scats.com.au/.

[133] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a Very Large Web Search Engine Query Log. In *Proceedings of the SIGIR Forum*, 1999.

[134] Skyhook Wireless. http://www.skyhookwireless.com/howitworks/.

[135] Alan Jay Smith. Sequentiality and Prefetching in Database Systems. *ACM Transactions on Database Systems*, 1978.

[136] Alan Jay Smith. Cache Memories. *ACM Computing Surveys*, 1982.

[137] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking*, 2005.

[138] Wei Wang Vikram Srinivasan and Kee-Chaing Chua. Trade-offs between Mobility and Density for Coverage in Wireless Sensor Networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2007.

[139] Karthikeyan Sundaresan and Sampath Rangarajan. On Exploiting Diversity and Spatial Reuse in Relay-enabled Wireless Networks. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008.

[140] Hwang Tae-Hyun, Joo In-Hak, and Cho Seong-Ik. Detection of Traffic Lights for Vision-Based Car Navigation System. *Advances in Image and Video Technology*, 2006.

[141] Stephen P. Tarzia, Peter A. Dinda, Robert P. Dick, and Gokhan Memik. Indoor Localization without Infrastructure Using the Acoustic Background Spectrum. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2011.

[142] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael A. S. Potts. Information Re-retrieval: Repeat Queries in Yahoo's Logs. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[143] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing Search via Automated Analysis of Interests and Activities. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[144] Arvind Thiagarajan, James Biagioni, Tomas Gerlich, and Jakob Eriksson. Cooperative Transit Tracking Using Smartphones. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[145] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[146] K. Thompson. UNIX Time-Sharing System: UNIX Implementation. *Bell System Technical Journal*, 1978.

[147] Joe Touch. Defining High Speed Protocols: Five Challenges and an Example That Survives the Challenges. *IEEE Journal on Selected Areas in Communications*, 1995.

[148] U.S. Department of Transportation. *IEEE 1609: Family of Standards for Wireless Access in Vehicular Environments (WAVE)*, 2006.

[149] J. van Leersum. Implementation of an Advisory Speed Algorithm in TRANSYT. *Transportation Research Part A: General*, 1985.

[150] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gün Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *Proceedings of the Workshop on the Economics of Peer-to-Peer Systems (NetEcon)*, 2003.

[151] Yiyan Wang, Yuexian Zou, Hang Shi, and He Zhao. Video Image Vehicle Detection System for Signaled Traffic Intersection. In *International Conference on Hybrid Intelligent Systems (HIS)*, 2009.

[152] Reinhold Weicker. Dhrystone: A Synthetic Systems Programming Benchmark. *Communications of the ACM*, 1984.

[153] Yinglian Xie and David O'Hallaron. Locality in Search Engine Queries and its Implications for Caching. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2002.

[154] Yelp. `http://www.yelp.com/`.

[155] Jeonghee Yi, Farzin Maghoul, and Jan Pedersen. Deciphering Mobile Search Patterns: A Study of Yahoo! Mobile Search Queries. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2008.

[156] Pei Zhang and Margaret Martonosi. CA-TSL: Energy Adaptation for Targeted System Lifetime in Sparse Mobile Ad-Hoc Networks. *IEEE Transactions on Mobile Computing*, 2010.

[157] Peng Zhou, Tamer Nadeem, Porlin Kang, Cristian Borcea, and Liviu Iftode. EZ-Cab: A Cab Booking Application Using Short-Range Wireless Communication. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2005.