Intelligent Transportation Systems Leveraging Next-Generation Mobile Devices, Sensors, and Networks

by

Jason Hao Gao

S.M., Massachusetts Institute of Technology (2013)
S.B., Harvard University (2010)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2016

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
August 31, 2016

Signature redacted

Certified by ... .
Li-Shiuan Peh
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Certified by ....
Anantha P. Chandrakasan
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by ....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Intelligent Transportation Systems Leveraging Next-Generation Mobile Devices, Sensors, and Networks

by

Jason Hao Gao

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Urban transportation is becoming increasingly intelligent and connected, with the potential for high societal, economic, and environmental impact as it changes the way we work and live in cities. Mobile apps today already provide navigation, transit prediction, mobility-on-demand, and other transportation services. Other urban transportation challenges, such as managing traffic congestion with high granularity and wide coverage, accessing real-time transportation and city information on-the-go, and deploying driver-less vehicles at scale, are still difficult to address pervasively because existing approaches require costly and slow-to-deploy infrastructure.

Our goal is to leverage the technological and marketplace forces of the mobile revolution to build and rapidly deploy pervasive, widespread, infrastructure-less intelligent transportation systems (ITS) that can address the needs of future smart cities. This thesis presents fully-integrated hardware and software systems with working, phone-based prototype deployments in cities. By focusing on pushing new technologies into the device rather than infrastructure, we can realize future ITS for smart cities more rapidly. Together, these systems enable a foundation for resilient, next-generation ITS apps that blur the line between city and software.

In the first part of this thesis, we observe the trend of increasingly diverse and varied wireless communications interfaces available on mobile phones, and design and build a prototype of an 802.11p radio that is suited for the power and size constraints mobile devices, allowing them to communicate directly with each other without routing through a router or cellular network. Our evaluation shows reductions in power consumption of 47-56% compared to an off-the-shelf 802.11p radio, and a significantly reduced system footprint, showing that 802.11p can be integrated as a future wireless communications interface on mobile devices.

We then propose and design a future ITS application that leverages device-to-device (D2D) communications to enable highly granular, widespread traffic management in cities: RoadRunner. We evaluate RoadRunner with both simulation studies and an experimental deployment on real vehicles to show that it achieves fine-grained traffic management and reduces traffic congestion, while eliminating the need for the

costly and coarse-grained infrastructure of existing traffic management systems.

In the second part of this thesis, we observe that mobile computing performance is improving rapidly, and propose that future ITS can eschew the traditional client-server approach and instead leverage the heavy-duty computation and D2D communications on the devices to improve user experience. We propose and design a suitable programming model and framework that seamlessly ties together device-centric computation and communications, allowing mobile app develops to easily develop applications in this proposed paradigm. We build and evaluate this programming framework, DIPLOMA, and an example ITS application on top of it, and demonstrate order-of-magnitude improvements in responsiveness/latency and reduced dependence on infrastructure-centric cellular networking.

In the final part of this thesis, we observe that mobile sensing is evolving rapidly and incorporating different sensing modalities. We propose that future ITS can use new sensors, such as laser distance sensors, by leveraging heavy-duty mobile computing performance, and design a low-cost laser distance sensor on a mobile phone. We build and evaluate our laser distance sensor in real-world conditions and on autonomous vehicles, and show that our prototype achieves performance suitable for collision avoidance for driver-less vehicles operating at up to 15-18 km/h, costs a fraction of the cost of other comparable laser distance sensors, and straightforwardly leverages improvements in mobile computing performance.

Thesis Supervisor: Li-Shiuan Peh
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Anantha P. Chandrakasan
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I give my sincerest thanks to the following people:

Professor Li-Shiuan Peh, Professor Anantha Chandrakasan, and Professor Daniela Rus, for illumination, advice, and support, for which I am immeasurably grateful.

Amy, Anirudh, Bhavya, HaoQi, Huayong, Kostas, Manos, Pablo, Pilsoon, Seth, Sunghyun, Suvinay, Tushar, and Woo-Cheol, for camaraderie and help in lab.

Professor Randall Davis, Janet Fischer, and the EECS staff, for patient guidance.

Maria Rebelo, for invaluable assistance with practical matters.

The many volunteers who helped with experiments.

Collaborators, from all around the world.

Frances, for unwavering encouragement.

Friends, for convivial respite.

Family, for faithful support.

Contents

1 Introduction **12**

1.1 Capabilities of modern smartphones 13

1.2 Future transportation challenges 14

1.2.1 Thesis contributions 16

2 Background **19**

2.1 Trends in vehicular technology 19

2.1.1 Vehicular Processing 19

2.1.2 Vehicular Sensing 20

2.1.3 Vehicular Communications 21

2.2 Trends in smartphone technology 22

2.2.1 Smartphone Processing 22

2.2.2 Smartphone Sensing 23

2.2.3 Smartphone Communications 25

2.2.4 Summary 29

3 Device-centric communications: Enabling highly-responsive ITS **31**

3.1 Introduction 32

3.2 Background 34

3.2.1 Phone Communications Circuits 34

3.2.2 802.11p Compatibility with 802.11a 34

3.3 System Prototype Design 36

3.3.1 FPGA Subsystem 36

3.3.2 Phone-FPGA Interface 37
3.4 Evaluation 38
3.4.1 RoadRunner Evaluation 39
3.4.2 SignalGuru Evaluation 41
3.4.3 Power Reduction Summary 42
3.5 Summary 42

4 Device-centric communications: Enabling pervasive traffic management 44
4.1 Introduction 44
4.2 Design 48
4.3 Implementation 50
4.3.1 Practical considerations 52
4.4 Deployment of RoadRunner Prototype 53
4.4.1 Request fulfillment offload 57
4.4.2 Request fulfillment time 57
4.4.3 Request fulfillment rate 58
4.4.4 Reroute notice time 59
4.4.5 System responsiveness 60
4.4.6 Cloud access offload 61
4.5 Large-Scale Simulation 61
4.5.1 Simulation setup and parameters 63
4.5.2 Simulation results 64
4.6 Related Work 65

5 Device-centric processing: Enabling pervasive computing for ITS 67
5.1 Introduction 68
5.2 The Design and Semantics of DIPLOMA 69
5.2.1 The Virtual Core layer (VCore) 70
5.2.2 The DIPLOMA Shared Memory layer (DSMLayer) 72
5.2.3 Snoopy and Resilient Cache Coherence (SRCC) 73

5.3 DIPLOMA Implementation 74
5.3.1 DIPLOMA's API 74
5.3.2 Prototype Design 75
5.3.3 Practical Considerations 76
5.4 Evaluating DIPLOMA 77
5.4.1 Benchmark App 77
5.4.2 Panoramio-like App 81
5.4.3 Simulation studies 83
5.5 Related Work 85

6 Device-centric sensing: Enabling pervasive autonomy for ITS 87
6.1 Introduction 88
6.2 Background and related works 91
6.3 Rangefinder Design 93
6.3.1 Challenges 94
6.3.2 Design Criteria and Characteristics 94
6.3.3 Ambient Light Rejection 97
6.3.4 Eye-Safety 98
6.3.5 Laser Line Localization 100
6.3.6 Calibration 100
6.4 Performance Evaluation 102
6.4.1 Outdoor Distance Sensing Evaluation 103
6.4.2 Obstacle Detection Evaluation 104
6.5 Full System Demonstration 107
6.5.1 Integration with vehicular platform 108
6.5.2 Demonstration behaviors 109
6.6 Discussion and Conclusion 110
6.6.1 Smartphone camera 110
6.6.2 Smartphone processor 112
6.6.3 Security 112

6.6.4 3D distance ranging and multi-user ranging 113
6.6.5 Conclusion 113

7 Conclusion **114**

7.1 Thesis Summary 114
7.2 Perspective and lessons learned 115
7.3 Future work 117
7.3.1 Security, privacy, and trust 118
7.3.2 Policy and regulatory considerations 119
7.3.3 Additional ITS services 119
7.3.4 New sensors 120
7.3.5 Modular hardware 120
7.3.6 New pervasive devices 121
7.4 Final remarks 121

List of Figures

2-1 Block diagram of Snapdragon 820 mobile System-on-Chip [48]. 23

3-1 RF front-end modules (FEMs) on the Apple iPhone 4. 32
3-2 Proposed interfacing of the 802.11p LEES single-die solution with existing phones' WiFi chipset. 35
3-3 FPGA system diagram. 38
3-4 Snapshot of the system prototype. 39
3-5 Average power consumption of RoadRunner V2V exchanges with and without adaptive power control. 41
3-6 Average power consumption of SignalGuru broadcasts with and without adaptive power control. 42

4-1 Pseudocode for RoadRunner. 49
4-2 Setup in each vehicle. 54
4-3 Map of deployment regions. 54
4-4 Deployment routes. Controlled regions have capacity shown. 54
4-5 RoadRunner deployment measurements 57
4-6 Map of Orchard Road and intersections for simulation. 64
4-7 RoadRunner Orchard Road simulation measurements 64

5-1 Walkthrough example of SRCC for two writes to VCore 5. Only VCores 3, 4 ,5 are detailed for clarity. 74
5-2 Completion rate, latency and power comparison of SMCloud and DIPLOMA in Pedestrian Deployment 78

5-3 Number of cloud accesses 81
5-4 Completion rate and latency of DIPLOMA in simulation. 84

6-1 Smartphone LDS with major components labeled. United States quarter for size comparison. 89
6-2 Total processing time per frame by our app versus image resolution. . 96
6-3 Geometry of angle to object. The angle θ is calculated from the y position of the laser illumination in the image. 97
6-4 Block diagram of hardware and software components. 98
6-5 Right: input image. Left: algorithm output. 99
6-6 Total processing time per frame vs. interpolation. 100
6-7 Calibration readings and 1/x curve fit. 101
6-8 Calibration error after 1/x curve fit. 102
6-9 Experimental setup of Smartphone LDS vs. XV-11. 103
6-10 Maximum range distance comparison. 104
6-11 Total error comparison. 105
6-12 Vehicular mount for system and experimental setup. 106
6-13 Diagram of obstacle avoidance scenarios. 107
6-14 Smartphone LDS mounted on a self-balancing scooter. 108
6-15 Video frames showing robot following a person. 110
6-16 Video frames showing robot avoiding an obstacle. 111

List of Tables

3.1 802.11p vs. 802.11a specifications. 35

5.1 DIPLOMA API Methods 75
5.2 Panoramio-like app latencies over 3G 83
5.3 Panoramio-like app latencies over 4G 83
5.4 Simulation settings 84

6.1 Smartphone LDS Characteristics 90
6.2 Smartphone LDS Cost Breakdown 90
6.3 Stopping Distance and Corresponding Speed 107

Chapter 1

Introduction

Mobile phones are ubiquitous today in urban environments, and possess powerful processing performance, multi-faceted sensing, and varied communications radios. These capabilities are being increasingly leveraged to improve transportation and future urban mobility, and can aid in tackling critical future urban transportation challenges as cities become more crowded.

In parallel, transportation networks and services themselves are becoming increasingly intelligent and interconnected, resulting in intelligent transportation systems (ITS) and connected vehicles (CVs). Emerging standards for wireless communication between vehicles promise to provide the interconnectivity required for ITS and CVs, but technology adoption in a typically safety-critical and government-regulated industry such as automotive can be slow relative to the mobile phone industry, which sees rapid introduction of new technologies.

Leveraging the quick turnover of mobile devices and smartphones can enable faster adoption of new technologies to support next-generation ITS. Critically, the convergence of mobile device technologies and transportation technologies can pave the way for more rapid deployment of next-generation transportation systems. This thesis presents a vision of future ITS where the mobile phone is the critical enabling component. Mobile phones in an urban environment will collectively sense, communicate, and process data to enable next-generation urban transportation. In the following chapters, we present the development of next-generation mobile phone-based sensors,

networks, and applications to realize this vision, and leverage them to tackle several challenges looming over urban transportation today.

In this thesis, we will show systems, devices, sensors, algorithms, and protocols to support our central thesis: that we can leverage ubiquitous computing devices, such as mobile phones, instead of relying on infrastructure development, to enable pervasive future transportation applications and services.

1.1 Capabilities of modern smartphones

Smartphones are not just pervasive, but also increasingly powerful: a modern smartphone has several communications interfaces, multi-faceted sensing capabilities, and multiple processing cores, all in a single device. A typical smartphone will contain:

1. **Cellular radios:** These provide basic voice and data communications capability. 3rd generation (3G) cellular data is prevalent in many parts of the world, and 4G cellular data technologies such as Long-Term Evolution (LTE) already widely deployed in many cities, improving bandwidth and latency compared to 3G. 5G technologies are now being tested in research [1] and industry [2].

2. **Short-range wireless communications:** A modern smartphone contains multiple short-range wireless interfaces, including WiFi, Bluetooth, Bluetooth SMART, and NFC. Some of these interfaces can be used for communicating directly between two smartphones, or device-to-device (D2D).

3. **General purpose processors:** Modern smartphones can perform increasingly intensive processing, and commonly possess multi-core processors such as the Samsung Exynos [3], Apple A9 [4] and Qualcomm Snapdragon [5]. As transistor density and corresponding processing performance increases in phones, heavy duty processing on the device can now be realized.

4. **Multi-facted sensors:** Smartphones have many sensors, including cameras, location sensors (such as GPS), accelerometer, gyroscope, compass, multiple microphones, ambient light, and proximity.

These varied capabilities can be leveraged to tackle challenges in realizing next-generation intelligent transportation services.

1.2 Future transportation challenges

Future Intelligent Transportation Systems (ITS) will require devices across entire cities to be interconnected, and ITS services will collect and provide real-time information to people, vehicles, and resources on the move. Smartphones are already ubiquitous in cities today, and people are accustomed to using them to access transportation services such as real-time mass transit scheduling and tracking, navigation, and mobility-on-demand (e.g. Uber, Lyft, GrabCar, and other phone-based car booking apps). In this thesis, we explore how the next-generation of smartphones will drive future ITS in the following scenarios:

1. **Real-time information sharing on the go.** With the prevalence of and policy focus on multi-modal transportation in urban areas [6], future ITS will need to provide relevant transportation information on-demand and quickly, while nimbly responding to changes in a highly dynamic environment with many kinds of vehicles and resources, and people are constantly in motion across multiple modes of transport. These ITS systems demand ultra-fast wireless communications at long ranges and high mobility, a challenging scenario for existing wireless communications standards, especially within the size and power constraints of mobile devices.

2. **City-wide fine-grained traffic management,** whether to mitigate and manage traffic congestion or to control emissions and the accompanying environmental impact, will be necessary to sustainably provide and improve the transportation efficiency of urban mobility. Existing traffic management schemes rely on physical infrastructure such as gantries and tollbooths, which are slow and costly to deploy, and prohibitively so for fine-grained, pervasive deployment. Other approaches to pervasive traffic management, such as deploying a

self-contained device in every vehicle, still require substantially upgrading existing cellular network infrastructure to address density, coverage, and handover issues to support millions of additional connected devices in a city.

3. **Wide adoption of driver-less vehicles.** The introduction of driver-less vehicles will require a ramp-up period on the way to pervasiveness. Two challenges impeding the speed of adoption are:

   (a) **The high cost of sensors for autonomy.** Autonomous vehicles (AVs) must sense and avoid obstacles and people, which is challenging in an unstructured urban environment. This is typically accomplished with expensive LIDAR sensors.

   (b) **The slow time scale for replacement of existing vehicles.** Vehicles are expensive purchases, with a slower replacement cycle than that of smartphones. Thus, there is a market for the retrofitting of autonomous capabilities onto existing non- or partially-autonomous vehicles [7, 8].

   Thus, low-cost sensing (both on and off-vehicle), processing, and real-time information sharing between sensors and vehicles will be necessary to speed the realization of autonomous vehicles while ensuring safe operation.

These scenarios highlight several challenges that have not been fully addressed by existing solutions. We foresee availability of the following capabilities in next-generation mobile phones, which will help address these challenges:

1. **Real-time information sharing:** Current networked mobile apps typically use a client-server model, where a thin front-end application on the mobile devices retrieves content and information from a back-end server that is responsible for information storage, retrieval, and processing. This model incurs a latency penalty as data must traverse the cellular network and public internet, perhaps several times, before relevant information can be presented to a user. With the availability of strong multi-core CPUs and GPUs on phones,

much of the processing can be moved closer to the devices, resulting improved responsiveness.

2. **Improved Device-to-Device communications:** The current D2D wireless standards on mobile phones are limited by their short range to static, largely indoors applications, and are not suitable for most transportation apps where individual nodes are moving around quickly and the network topology can be rapidly changing. Longer-range D2D standards (such as 802.11p) are currently restricted to non-mobile-phone applications such as vehicle-to-vehicle communications due to power and size constraints.

3. **New sensors:** Newer smartphones are already beginning to include a wider array of sensors, such as barometer, temperature, humidity, heart-rate, and time-of-flight autofocus assist, in addition to the ones mentioned earlier in Section 1.1. One key sensor that is not yet widely available in mobile phones, however, is depth sensing. Emerging depth sensors for mobile devices, such as Project Tango [9] and Structure Sensor [10], have been demonstrated, but have not seen wide adoption yet due to performance, power, cost, and size constraints. Another reason that these depth sensors have not been widely adopted in transportation is that they do not work outdoors, and laser distance sensors that do work well outdoors, such as LIDAR, are prohibitively expensive.

1.2.1 Thesis contributions

This thesis tackles these scenarios and challenges holistically by demonstrating fully-integrated hardware-software systems on working phone-based prototypes to address challenges in communications, processing, and sensing for next-generation ITS:

Device-centric Communications for ITS

We present D2D communications radios (Chapter 3) that work at high mobility and long range, and which can be embedded into pervasive devices such as smartphones

within their challenging constraints of power and size. We also present an ITS application that showcases the impact of D2D communications (Chapter 4) and a programming layer that leverages D2D communications to tie together processing and sensing for ITS (Chapter 5).

Device-centric Processing for ITS

We present a programming layer that eases the use of processing for distributed mobile ITS applications (Chapter 5), and leverages the strong processing performance of multi-core CPUs and GPUs available on smartphones. It utilizes D2D communications to enable programmers to easily build apps with real-time information sharing on the go, running across a network of pervasive mobile devices. We evaluate and demonstrate the capabilities of this programming layer in the context of a mobile app for urban imagery.

Device-centric Sensing for ITS

We present a smartphone-based laser distance sensor for ITS applications (Chapter 6). The sensor is low-cost enough to be pervasively used, whether embedded on self-driving cars, installed on autonomous drones, included in retrofit kits for older vehicles, or dispersed across an urban environment for widespread, city-scale sensing. This sensor can be coupled with other low-cost phone sensors (e.g. GPS), sophisticated processing for ITS, and widespread D2D communications for ITS, to provide contextual, multi-faceted information for next-generation ITS applications. We evaluate and demonstrate the capabilities of this sensor in the context of autonomous vehicles.

Further contributions

In the course of this thesis work, we also present contributions in developing novel algorithms, devices, and sensors, that are more broadly applicable:

- We extend a theoretical abstraction for mobile ad-hoc networks, Virtual Nodes,

to address and solve challenges that arise from assumptions that do not hold true in the real world; such challenges include unreliable wireless communications and intermittent network partitions and failures. We also present a real-world evaluation of these theoretical ideas as we translate them from theory to practice.

- We present a protocol for decentralized decision making in motion coordination for cars that could be extended for use in other contexts such as mobile robotics, and evaluate its effectiveness in achieving target densities of nodes in physical areas.

- We adapt concepts and protocols from networks-on-chip in computer architecture design and evaluate them in a novel context, mobile ad hoc networks, while overcoming obstacles like unreliable and unordered-delivery interconnects between cores and routing failures from mobile nodes going offline.

- We present a novel sensor and algorithm for distance sensing that can robustly track modulated laser illumination even in the presence of high ambient light. This algorithm could be used in additional applications such as outdoor motion tracking and visible-light communications.

These further contributions have potential for use in areas beyond intelligent transportation, such as robotics, computer architecture, computer vision, mobile ad-hoc networks, and distributed computing.

Chapter 2

Background

2.1 Trends in vehicular technology

2.1.1 Vehicular Processing

Modern vehicles contain as many as a hundred individual processors [11, 12], connected over the Car Area Network (CAN) bus. These processors are responsible for many discrete tasks within the vehicle, such as regulating the engine behavior via the Engine Control Unit (ECU), deploying airbags in the event of a collision, and operating the entertainment system. The tougher vehicular environment, with higher thermal and reliability requirements, results in the use of lower performance processors on older technology nodes [13, 14].

As vehicles become increasingly intelligent and include capabilities for advanced driver assistance systems (ADAS) such as adaptive cruise control to automatically brake to avoid collisions [15] and automatic lane following [16], to fully autonomous self-driving capabilities [17], more processing performance is being put into vehicles [18]: Mobileye's EyeQ4 automotive processor is capable of 2.5 Teraflops of processing performance [19], Nvidia's Drive PX automotive processor is capable of 2.3 Teraflops [20], and Intel plans to bring its processors to cars, "delivering up to a total of 100 teraflops of power efficient performance" [21].

2.1.2 Vehicular Sensing

Vehicles already depend on many sensors to measure many parameters of the vehicle itself necessary to operate, e.g. fuel flow, air intake, power steering, brake pressure, etc. Vehicles have also increasingly incorporated additional sensors to monitor the exterior world to support additional functionality:

- **Cameras**: Vehicles frequently incorporate rear-mounted cameras in order to provide an unobstructed view of the area behind the vehicle, to assist the driver when the car is being operated in reverse [22]. Forward-mounted cameras are also being introduced onto vehicles, such as Tesla cars [23], to support functionality such as automatic lane keeping and limited self-driving capabilities.

- **Radar**: Forward-looking radar is available in some vehicles to support adaptive cruise control and collision avoidance [24] by measuring the distance to cars or obstacles in front of the vehicle, and automatically adjusting the speed to maintain distance or automatically stopping the vehicle if necessary.

- **Sonar**: Sonar sensors measure distance to an object by measuring the time it takes for a ultrasonic acoustic pulse to be reflected from the object. Some vehicles incorporate sonar sensors, to alert the driver of unseen obstacles (e.g. behind the vehicle) and/or to support functionality such as automatic lane changing [25].

- **Laser distance sensors**: Various organizations developing self-driving cars have incorporated laser distance sensors such as LIDAR into their vehicles, including Google [26] and Uber [27]. These sensors provide a higher-resolution measurement of the distance from the vehicle to various objects around it than sonar can.

While cameras, radar, and sonar are commercially available on vehicles today for ADAS functionality, high-resolution laser distance sensors are not commercially widespread on vehicles. Laser distance sensors are emerging as a key sensor for future

vehicles, such as self-driving cars, which use them for collision avoidance, object detection and recognition, and localization and mapping [28, 29].

LIDAR is a laser distance sensor that works on the same principle as radar, but uses light from a laser. LIDAR has the advantage of high spatial resolution, allowing vehicles to discriminate between multiple types and sizes of obstacles, but its cost is prohibitive for pervasive deployment due to the need for high-speed circuitry for accurate time-of-flight ranging, high-powered laser diodes and photodetectors, and electromechanical components to scan the optics over the field of view.

LIDAR sensors work on the principle of Time-of-Flight (ToF). The sensor emits a very short, high-power laser pulse, and measures the time it takes to detect the reflection of the pulse from a distant surface. Commonly used LIDAR sensors on prototype self-driving vehicles include the SICK LMS 291 and Velodyne 3D LIDAR sensors [30].

2.1.3 Vehicular Communications

Communications for future vehicles is moving towards Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) wireless communications, collectively termed V2X. There is increasingly wide industry and government support, including programs in Europe (Car2Car [31]), Germany (simTD [32]), Japan [33], the United States [34], and Singapore [35]. Numerous V2X applications have been proposed or deployed, including apps in multimedia, safety, road pricing, and more [36]. The use of V2X is due to recognition of the need for fast, seamless communications on-the-go, and that existing wireless communications standards, such as WiFi or LTE cellular networks, do not suffice:

- WiFi and LTE's centralized architecture incurs higher latency, as communication between vehicles must traverse a base station; thus, they are unable to support the real-time response required for safety apps [37].
- Cellular data networks are increasingly overloaded, metered, and capped [38]. Even additional capacity will be overtaken by increased demand from: 1) Use

of LTE in new vehicles and home broadband; 2) Higher screen resolutions that increase demand for high-definition content.

- WiFi service is often available at subway stations, bus stops, etc., but its short range makes it difficult to support uninterrupted communications with high mobility clients [39].

Heterogenous networks (HetNets) promote a vision of achieving pervasive Internet communications by leveraging varying network technologies and adhoc networks. Related works on vehicular adhoc networks (VANETs) in particular address topics such as seamless 3G communications across fleets [40], vehicular routing [41], context-aware content delivery [42], and mobile IP optimizations [43], but work on HetNets has not focused on hosting services and applications directly on the mobile devices, within the VANET. Emerging V2X promises significant benefits for both HetNets, and for next-generation ITS apps that leapfrog the need to access Internet services:

Ultra-fast response times: Low latency vehicular communications is critical in transportation apps, especially safety applications.

Location-based services: V2X complements the location-aware nature of ITS apps, and naturally connects nearby nodes with relevant information.

The adoption of V2X technology, however, depends on the deployment of V2X-enabled devices in vehicles and roadside access points, which hinges on customer renewal rate of vehicles and transport authorities' policies. This is slow compared to the renewal rate of consumer technology products, such as smartphones.

2.2 Trends in smartphone technology

2.2.1 Smartphone Processing

Processing performance of smartphones and mobile devices is already very high today, and continues to increase exponentially in performance and energy efficiency [44, 45]. Furthermore, the processors in modern smartphones are comprised of multiple identical cores (multi-core) for executing more than one processing thread in parallel, and

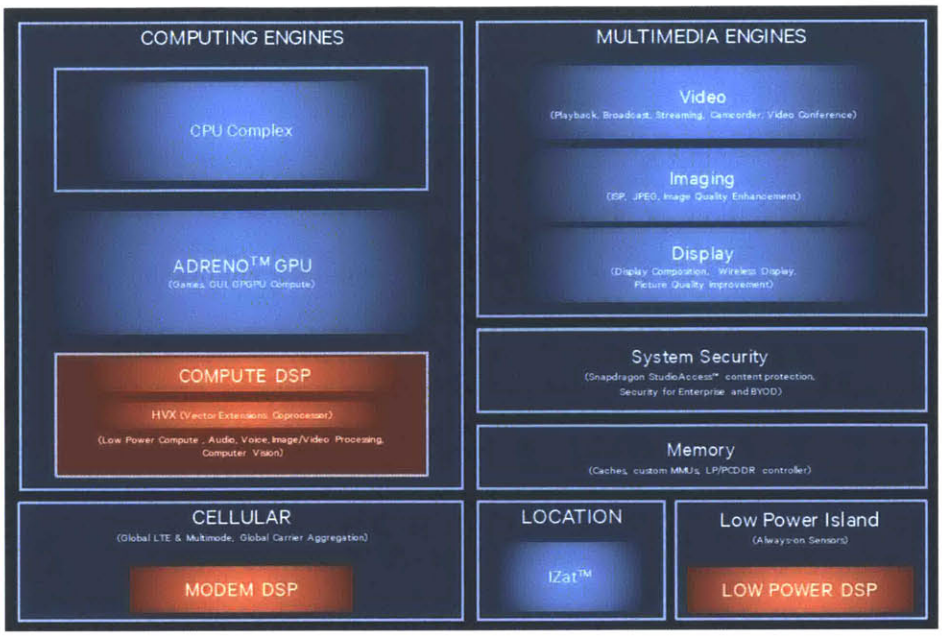


Figure 2-1: Block diagram of Snapdragon 820 mobile System-on-Chip [48].

often include additional heterogenous processing cores as well, onto a single System-on-Chip (SoC). ARM's big.LITTLE architecture combines heterogenous CPU cores onto a single SoC, and has already been implemented in smartphone SoCs [3]. Graphics Processing Units (GPU) are included on SoCs to accelerate graphics operations, e.g. for games, user interface compositing, and video rendering, and are also being leveraged by mobile apps to accelerate performance of highly parallel processing tasks such as matrix math and image filtering operations [46, 47].

Thus, a typical smartphone SoC actually contains several different kinds of processing capabilities, via the CPU cores, GPU cores, and DSP cores, that can be harnessed to provide heavy-duty processing on the device, as shown in Figure 2-1, which illustrates the different processing engines available in a Qualcomm Snapdragon 820.

2.2.2 Smartphone Sensing

Smartphones also contain a multitude of sensors; commonly known ones include GPS, accelerometer, and the camera. These mobile sensors have been widely leveraged for a variety of applications, such as mobile crowdsourcing [49] and mobile healthcare [50].

Below, we describe several sensors found on smartphones, and some of their uses for transportation applications:

- **Location**: Location sensing through methods such as the Global Positioning System (GPS) allow smartphones to localize themselves within a few meters, but requires a clear view of the sky [51] and consumes significant power [52]. Other methods such as Wi-Fi triangulation and cellular tower triangulation consume less power than GPS, but have poorer accuracy: tens of meters for Wi-Fi [53], and hundreds to thousands of meters for cell tower triangulation [54]. Recently, organizations have begun deploying Bluetooth Low Energy (BLE) beacons for fine-grained localization of devices in GPS-denied environments (e.g. indoors or underground) [55]. Location sensors have been used in ITS apps such as traffic monitoring [56] and user activity context detection [57].

- **Camera**: Smartphones typically include one or more cameras, which allow them to capture photos and videos. Cameras have been used in ITS apps such as traffic light prediction and road advisories [58], bus capacity prediction [59], and vehicular visible light communication [60].

- **Inertial and orientation**: Together, the accelerometer, gyroscope, and magnetometer in a smartphone form an Inertial Measurement Unit (IMU). The accelerometer measures linear acceleration of the phone in 3 dimensions, the gyroscope measures rotation rate around 3 axes, and the magnetometer measures magnetic field strength and orientation. Inertial sensors have been used for ITS apps such as transportation activity and mode detection [61] and indoor positioning [62], and outdoor positioning [63].

- **Barometer**: Barometers are a relatively recent addition to smartphone sensing, and measure air pressure, which can be used to infer height above sea level. They have very good relative accuracy, and have been used for transportation context detection [64], improving indoor navigation via floor-change detection [65, 66], and accelerometer compensation [67]. One advantage of barometers over other sensors is that they consume very little power [64].

- **Proximity**: Smartphones often include short-range proximity sensors that can measure distances along a single axis, to perform simple context-sensitive actions such as turning the screen off when the user places the phone next to his head. Recently, smartphones have included additional proximity sensors on the back of the phone to assist with camera autofocus [68].

- **Microphone**: Smartphones typically include multiple microphones, which have been used for applications beyond supporting the basic functionality of phone calls. Examples of ITS applications include crowd counting [69], traffic condition monitoring [70], and accident detection [71].

2.2.3 Smartphone Communications

The wireless communications interfaces commonly available on phones today can be divided into two categories: long-range cellular communications and short-range wireless communications. We will also discuss the emerging class of device-to-device wireless communications not yet prevalently used on smartphones, but which holds promise for enabling very low-latency wireless communications without requiring additional physical infrastructure to be deployed.

Long-range cellular communications

Long-range cellular communications permits phones to connect to the Internet and other services provided by wireless Internet Service Providers (ISPs), also known as wireless or mobile carriers in the context of mobile phones. Long-range cellular communications depends on physical infrastructure, specifically cellular base stations, to be deployed. The "cell" in cellular refers to the range of a cellular base station, which can have a radius of 10s of kilometers. Due to demand for mobile data outstripping capacity even with evolving technologies, and the need for more efficient use of wireless spectrum, however, there is an increasing focus on smaller cell sizes, leading to terms like micro-cell, pico-cell, and femto-cell [72, 73].

Several generations of standards for cellular communications have been deployed.

Modern smartphones include 4th generation cellular communications technology (4G), and typically include hardware to remain compatible with preceding 3G, 2G, and 1G wireless communications standards. The technology standard that has become widely adopted by 4G phones is Long Term Evolution (LTE), which in real-world use, provides data rates of several tens of Megabits per second and round-trip latencies around 70 milliseconds [74].

The next-generation 5G standard for cellular communications is emerging, but it is still being developed and it is not certain what technologies will be included in it from among several competing technologies [72]. Technologies being considered for use in 5G networks include millimeter wave with beamforming, massive / multi-user multiple-input multiple-output (MU-MIMO), device-centric architecture, and femtocells [75]. The infrastructure-centric nature of cellular networks results in a long lead-up to pervasive deployment of new technologies: despite its commercial introduction in 2009, 4G is still not as prevalent as 3G today in 2016 [76].

Short-range wireless communications

Phones also contain short-range wireless communications interfaces for various purposes, including communications to the Internet and/or communications with peripheral devices such as headsets, keyboards, and wearable devices. Well-known and widely-used standards are WiFi (e.g. 802.11a/b/g/n/ac), Bluetooth, and Near Field Communication:

- **WiFi:** WiFi [77], most commonly used in *infrastructure mode*, typically provides network communications, including to the Internet, and can support higher data rates and lower latency than long-range cellular communications such as LTE, but it has a shorter range. WiFi can provide tens of meters of range and typical data rates of tens to hundreds of Megabits per second [78]. Similarly to cellular networks, WiFi in infrastructure mode depends on the deployment of physical infrastructure, WiFi access points (APs). WiFi in ad-hoc mode and WiFi Direct do not require access points to be deployed, and are described in further detail in Section 2.2.3.

- **Bluetooth:** Bluetooth [79] is oriented around a master-slave paradigm, as it is primarily used for wirelessly connecting peripherals such as headsets, speakers. In Bluetooth networks, one device is a master and up to seven slave devices can connect to it, and typical data rates are a few Megabits per second [80]. Devices are paired through either a manual setup procedure, or more recently, via Near Field Communication (described below). Bluetooth Smart, also known as Bluetooth 4.0 or Bluetooth Low Energy, is a more recent variant that reduces power consumption and allows for more peripherals to be connected at once, at the expense of lower data rates and higher latency [81].

- **Near Field Communication (NFC):** NFC is a short range, low data rate communications standard limited to distances of a few centimeters between two devices, and a relatively low maximum data rate of 424 kbit/s [82]. It is often used for low data rate tasks like out-of-band pairing for Bluetooth or Wi-Fi, reading an identifier from a physical object [83], or mobile payments [84].

Direct Device-to-Device (D2D) communications

There is also an emerging class of smartphone communications that focuses on facilitating communications directly between phones, without being routed through an intermediate access point or cellular tower. Some wireless standards capable of supporting device-to-device (D2D) communications for phones (analogous to V2V for vehicles) have been available for many years, such as ad-hoc WiFi and Bluetooth. Mobile operating systems have also begun to provide support for location-aware device-to-device communications and service discovery through Bluetooth, WiFi Direct [85], and iOS Multipeer Connectivity [86].

LTE Direct is a promising new D2D technology, but as it leverages LTE infrastructure, it requires modifications to the LTE base stations which may hinder adoption. Next-generation long-range cellular communications, such as 5G networks, have proposed capabilities for device-to-device communication like LTE Direct, but as mentioned before, it is unclear when 5G will be adopted and in what form and

capabilities.

WiFi Direct has seen good adoption among Android devices: video sharing, file sharing, and multi-player gaming apps have started to leverage WiFi Direct. Applications that benefit from the faster response times of D2D communications typically gather user input and sensor data from nearby phones, perform processing in-situ, and output results and user interface updates with high responsiveness. However, WiFi Direct uses the existing WiFi hardware, and thus only works for short-range, low mobility scenarios: WiFi communications, in general, is challenged in long-range or high-mobility scenarios [87].

Thus, D2D interfaces available today are limited to short-range, low mobility scenarios due to software and hardware design. There are constraints on the number of devices, mobility of devices, and on the ease of setup and interoperability, making them unsuitable for the needs of highly responsive, mobile, and pervasive ITS apps. We've identified several key roadblocks for existing D2D in tackling ITS scenarios below:

- **Centralized topology:** WiFi Direct facilitates easier setup of D2D networks, but one device must serve as an access point (the *group owner*) and all other devices must route communications through it, resulting in increased latency and a brittle, centralized architecture that does not support highly mobile networks with rapidly changing topologies. This largely limits WiFi Direct apps to close-range, static deployments between a handful of phones. The emerging LTE Direct [88] standard similarly requires the use of the LTE base station as a centralized coordinator. Bluetooth is also limited to star topologies, and requires a more cumbersome pairing process than WiFi Direct.

- **Lack of seamless communications:** WiFi Direct and Bluetooth require user interaction to initiate connections between devices, limiting their usefulness in ITS scenarios where nodes may rapidly come and go in the background as users move around urban environments.

- **Lack of pervasive support:** The existence of several D2D standards has

fragmented platform support. Android does not support ad-hoc WiFi [89], while iOS has limited support. Android supports WiFi Direct, while iOS does not. iOS Multipeer Connectivity combines Bluetooth and WiFi, but is limited to iOS devices. Different mobile operating systems, and even individual devices with the same OS, often do not support the same Bluetooth profiles. LTE Direct is not available yet, and depends on support from cellular service providers and cellular tower infrastructure.

These limitations render the aforementioned standards unsuitable for ITS scenarios, characterized by highly mobile nodes with rapidly changing topologies that must communicate across long distances, quickly and seamlessly in the background. In contrast, vehicular V2V communications does have well-adopted standards which meet the needs of those ITS scenarios, and is interoperable across many different brands and models of vehicles, road-side infrastructure, and mobile hardware.

2.2.4 Summary

Smartphones are pervasive in many cities, and the fast upgrade cycle of smartphones means new technologies are introduced very quickly in phones. Smartphone processing performance is rapidly scaling, and smartphone processors are typically manufactured using newer process technologies than vehicular processors, offering an advantage in cost. Furthermore, smartphones and other mobile devices are including more and more diverse sensing modalities, many of which now overlap with sensors on vehicles, such as radar [90], sonar [91], and more.

Smartphone capabilities are also increasingly more available to vehicles: car manufacturers are designing vehicles with the capability to interface to phones, offering features such as integrated information sharing and communications with mobile devices [92], built-in dashboard docks for smartphones [93, 94], and there exist aftermarket devices to bring cellular communications and telematics to existing vehicles [95].

As phones and vehicles become more interconnected and interdependent in future transportation systems, and vehicular and smartphone technologies converge, we can

realize device-centric ITS, in which new ITS technologies and improvements are deployed via new mobile devices rather than via new vehicles or new infrastructure, resulting in quicker and more widespread adoption.

Chapter 3

Device-centric communications: Enabling highly-responsive ITS

This chapter contains joint work with Pilsoon Choi, Nadesh Ramanathan, Mengda Mao, Shipeng Xu, Chirn-Chye Boon, Suhaib Fahmy and Li-Shiuan Peh, presented at the International Symposium on Low Power Electronics and Design (ISLPED) in August of 2014, titled "A case for leveraging 802.11p for direct phone-to-phone communications".

This chapter presents a mobile 802.11p wireless radio targeted for enabling long-range Device-to-Device (D2D) communications between mobile devices that reduces communications latency and reduces the need for long-range communications across cellular networks. The design utilizes a new circuit manufacturing process to reduce the area footprint and power consumption of the 802.11p radio, making it suitable for integration into a mobile device. We evaluate the improvement in power reduction for two example ITS applications and show that long-range D2D communications can be realized as a future wireless communications interface to enable highly-responsive ITS applications. In the following chapter, we motivate the advantages of D2D for ITS in detail with a traffic management system that leverages D2D communications to provide fine-grained, wide coverage of traffic management as city-scale.

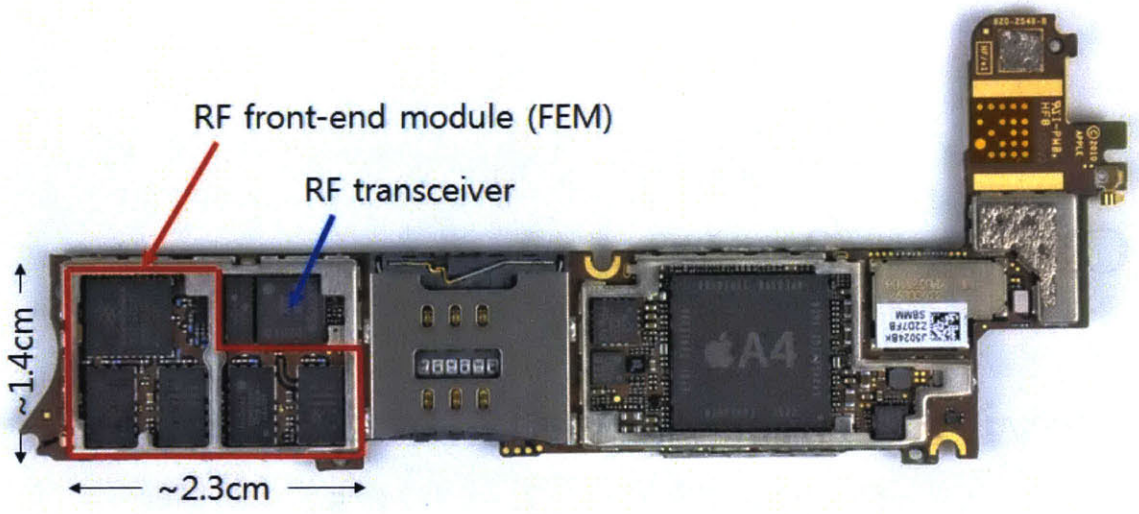


Figure 3-1: RF front-end modules (FEMs) on the Apple iPhone 4.

3.1 Introduction

Vehicle-to-vehicle (V2V) communication is a form of D2D communication that enables vehicles to wirelessly communicate with each other, and has been burgeoning with the adoption of the IEEE 802.11p DSRC standard around the world [96, 31]. Numerous V2V applications in the transportation domain have been proposed or deployed, such as mobile multimedia, safety, road pricing, and others [36]. These applications leverage the high mobility, long range and fast response times of 802.11p for next-generation transportation applications. 802.11p's increased transmit power enables longer range communications, but the high power consumption of 802.11p radios has, until now, precluded their integration into non-vehicular mobile devices[1].

In this chapter, we demonstrate the feasibility of realizing 802.11p on phones by bringing together materials, devices, circuits, and systems researchers. This development opens up D2D communications to a much larger class of applications, with mobile devices on pedestrians, passengers, and drivers now interconnected at low latency and high bandwidth, enabling highly interactive mobile ITS.

Among several building blocks for a communications system, the RF front-end is one of the most critical, with III-V semiconductor devices (e.g. GaN, GaAs, InGaP)

[1]The recently-released Qualcomm Snapdragon Automotive Solutions support DSRC for short-range vehicular safety detection, but not 802.11p and its extended range with high transmit power.

showing much better power density and efficiency than CMOS. Figure 3-1 (photo from [97]) shows multiple RF front-end modules (FEMs) for a variety of standards in an Apple iPhone 4; together, these occupy a large portion of real estate. In addition, each FEM includes multiple semiconductor dies within it, further increasing area footprint, power, as well as cost.

In our work, we leverage a unique process, the LEES (Low Energy Electronics Systems) process, where both CMOS and III-V semiconductor devices can be fabricated on a *single* die. This allows the use of the most suitable III-V devices grown on top of a conventional CMOS device, interfaced via metal layers. Such single-die integration offers the superior performance required by 802.11p specifications at the small form factor and within the tight power budget of a smartphone implementation. In Section 3.2, we show how a newly fabricated FEM can integrate into the existing communications subsystem circuitry on a phone.

We demonstrate compatibility with existing phones by emulating an 802.11p baseband on FPGA (using a modified 802.11a baseband) and interfacing the FPGA with the fabricated 802.11p transmitter. Our system prototype is a transmitter chain consisting of the designed front-end circuits in standard CMOS and GaN technologies, a baseband processor in an FPGA board interfaced to an Android smartphone through USB, all 802.11p compliant. Application-level adaptive control of the radio's transmit power through a gain control interface means the Android application can tune the radio's transmit power (and thus its power dissipation) to match actual desired D2D communication distance. This joint hardware-software power optimization enables substantial further power reduction, allowing the prototype to meet the aggressive smartphone power budget.

3.2 Background

3.2.1 Phone Communications Circuits

A typical smartphone incorporates several two-way communications radios, including WiFi (IEEE 802.11a/b/g/n/ac), Bluetooth, and the cellular radios. The cellular radios in mobile phones available today do not support direct device-to-device (D2D) communications, and only communicate with the cellular base stations that coordinate access to the medium. WiFi Direct is a recent standard that allows D2D communications between mobile phones, and thus enables networks with star topologies, but not mesh or full peer-to-peer topologies. Ad-hoc WiFi is a pre-existing standard that allows for direct D2D communication without needing to appoint one of the devices as a centralized controller or access point, but is not widely supported among the major mobile operating systems, and thus requires kernel modifications.

Each radio typically contains a PHY (physical layer) and MAC (medium access control) implemented in hardware, with upper MAC and higher networking layers implemented in software at the device driver, operating system and application level. Most of the building blocks of the communications subsystem within a phone are increasingly being integrated with current standard CMOS processes, except for the RF power amplifier. While there is significant ongoing circuit research targeting CMOS power amplifiers to enable higher level of integration of the entire communications subsystem, the intrinsic low power density and efficiency of current CMOS devices presents a tough challenge [98]. As shown in Figure 3-1, a power amplifier for each communication standard is still a separate chip fabricated using III-V technology which enables higher output power and efficiency, but worsens system form factor.

3.2.2 802.11p Compatibility with 802.11a

IEEE 802.11p DSRC is an emerging standard originally proposed for vehicle-to-vehicle (V2V) and vehicle-to-infra structure (V2I) communication, enabling truly distributed mesh D2D networking like ad-hoc WiFi. Table 3.1 compares the 802.11p DSRC

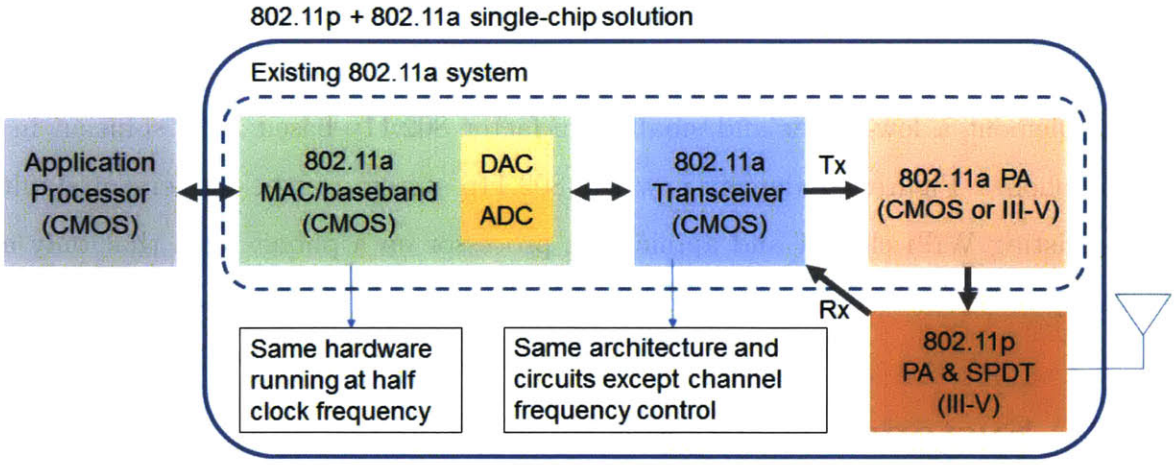


Figure 3-2: Proposed interfacing of the 802.11p LEES single-die solution with existing phones' WiFi chipset.

specification with 802.11a WiFi. 802.11p adopts the same OFDM modulation as 802.11a/g, but its time domain parameters are double those of 802.11a to mitigate highly mobile and severe fading vehicular environments. Thus, when we implement the digital baseband processor for 802.11p, we can use 802.11a WiFi hardware as is, but run it at half the clock frequency.

	802.11p	802.11a
User mobility	Vehicular (outdoor)	Personal (indoor)
Operating frequencies	5.85-5.925GHz	5.15-5.825GHz
Channel bandwidth	10MHz	20MHz
Max. output power	760mW	40/200mW
Data rate	3-27Mbps	6-54Mbps
Modulation	BPSK-64QAM	BPSK-64QAM
OFDM symbol duration	8us	4us
Guard time	1.6us	0.8us
Preamble duration	32us	16us
Subcarrier spacing	156kHz	312kHz

Table 3.1: 802.11p vs. 802.11a specifications.

However, with the increased transmit power and robustness necessary for longer range V2V communications, the high power consumption of 802.11p radios has precluded their use in mobile phones until now. The LEES process can resolve this issue through its novel CMOS/III-V integration technology [99] that can optimize high power density III-V devices' performance for specific applications and integrate them

with CMOS on a single die. This novel device, process technology, and circuit design, combined with adaptive gain control by the application software, make it possible to implement a low-power and small form-factor 802.11p-based D2D solution in a smartphone. Figure 3-2 depicts how the 802.11p system can be implemented with the existing WiFi chipset and application processor on a phone, such that only an 802.11p RF front-end chip needs to be added.

3.3 System Prototype Design

Our system prototype is a transmitter chain from phone to FPGA to the RF front-end, enabling 802.11p compliant signal transmission with application level gain control for power saving. A USB-Ethernet adapter is used to communicate between the Android phone and the FPGA. To interface the baseband processor on FPGA to our custom RF front-end, commercial DAC evaluation boards are used to feed analog I/Q signals into the RF transmitter. As mentioned in Section 2.2, all these digital and ADC/DAC components can be shared with existing WiFi communications circuitry, and a single RF front-end can readily support both 802.11a and 802.11p by slightly extending its maximum carrier frequency range from 5.875GHz to 5.925GHz. Thus, to demonstrate the feasibility of 802.11p implementation and its compatibility with existing WiFi solutions, we design and fabricate a CMOS+GaN RF front-end consisting of a CMOS transmitter and a GaN PA. The CMOS circuit is fabricated in 0.18um CMOS technology with a die area of $1.4mm^2$. The GaN circuit is fabricated in a 0.25um GaN-on-SiC process [100] with a die area of $1.28mm^2$. We then adapt an existing 802.11a baseband implementation to implement an 802.11p baseband on an FPGA, and interface the FPGA to an Android smartphone.

3.3.1 FPGA Subsystem

FPGAs provide an ideal platform for prototyping complex radio baseband implementations in real-time, offering high performance, low power, and portability, in comparison with other software radio platforms [101]. The FPGA platform performs

two vital functions in our setup: baseband processing and providing the interface between the application software on the Android phone and the analog/RF circuitry via the DAC. The FPGA system is implemented on a Xilinx XC5VLX110T FPGA on the XUPV5 development board. We have implemented the complete transmitter chain as described in Figure 3-3. Packets are transmitted from the Android handset, via the RRR Abstraction layer and Ethernet physical interface, into the 802.11p Airblue baseband on the FPGA. The resultant baseband output is passed through a digital low pass filter and scaled before delivery to the DAC circuitry.

The Airblue baseband [102] was originally designed for the 802.11a standard. Since the two standards are largely similar, we run the entire baseband design at half the clock frequency (10MHz) to achieve compatibility with the 802.11p standard. It is worth noting that for actual 802.11p deployment, more stringent output spectrum shaping is required than for 802.11a [103]. The Android handset can access two functions in the FPGA hardware: the packet generator and the gain control module. The packet generator is responsible for configuring parameters, buffering, and synchronizing, the baseband transmission. The gain control module allows the Android handset to directly control power settings on the RF front-end. The FPGA receives and decodes power control commands from handset, applying the appropriate settings at the front-end via a parallel pin interface. This enables power saving capability to be applied from the Android application software. We add a digital low pass filter to reduce the noise caused by the sampling effect within the 40MHz spectrum range.

The Asim Architect's Workbench (AWB) [104] is the development environment for hybrid hardware-software design. FPGA support is provided in AWB via the Logic-based Environment for Application Programming (LEAP) framework [105] that provides the Remote Request-Response (RRR) framework, an abstracted communication layer.

3.3.2 Phone-FPGA Interface

The Android smartphone is interfaced to the FPGA through a USB-Ethernet adapter connected via Ethernet to the FPGA and via USB On-the-Go (OTG) to the Android

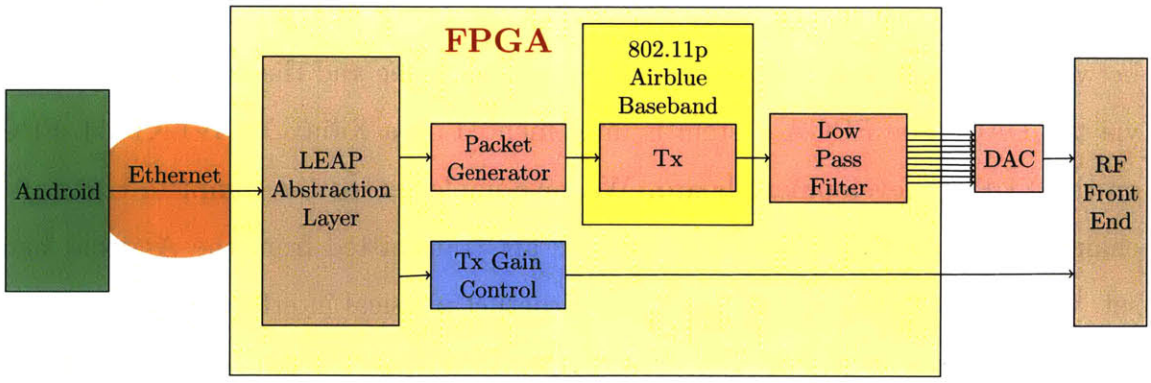


Figure 3-3: FPGA system diagram.

device.

In order to have the Android device recognize and enumerate the USB-Ethernet adapter, we recompiled the Linux kernel for the device to include the USB-Ethernet drivers for the particular ASIX AX88178 and SMSC 7500 chipsets in the adapters. We then loaded this kernel onto the phones, replacing the default kernel. This allowed the Android device to become a USB host and recognize the USB slave Ethernet adapters attached to it via the USB OTG cable.

3.4 Evaluation

Figure 3-4 shows the experimental setup of the system prototype, illustrating that a smartphone, an FPGA board and commercial DAC evaluation boards are interfaced to the designed CMOS and GaN PCB boards. An 802.11p compliant digital baseband implemented in the FPGA along with the TI dual 12-bit DAC, DAC2902, sampling at 40MHz, feeds the analog I/Q baseband signals into the CMOS transmitter. An Android application on the smartphone controls packet generation/transmission and RF gain.

Since the transmit mode dominates power consumption, we design and implement an entire transmitter chain to validate the LEES feasibility as well as potential power reduction through application-level adaptive power control (ALAPC). In addition, the DC power of the PA is more than 90% of the whole transmitter power with a

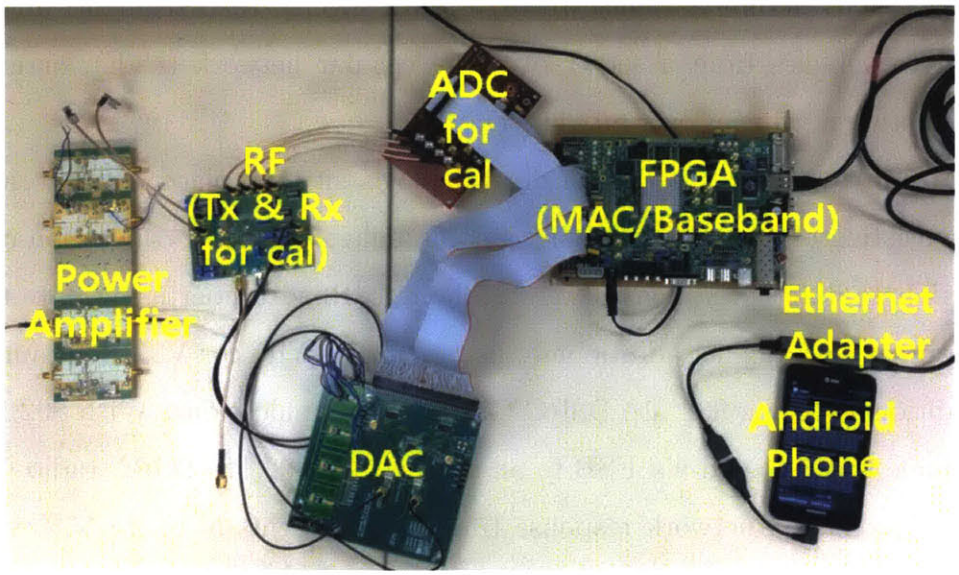


Figure 3-4: Snapshot of the system prototype.

complex modulation scheme like OFDM in 802.11p, since it requires back-off due to its high PAPR signals and the power efficiency is dramatically reduced as output power decreases from the saturation point. Thus, power management of the PA is crucial to fit the 802.11p front-end within a smartphone's stringent power budget.

In the following subsections, we demonstrate that ALAPC, combined with our GaN PA's improved power efficiency across all output power levels, can achieve dramatic power reductions. We cannot yet deploy our prototype system due to its complex system configuration, FPGA and DAC boards, and multiple power supplies for the transmitter and PA boards. However, we can use traces from prior deployments of two mobile apps which originally used off-the-shelf D2D communications, to estimate the potential system power savings that can be achieved if we replace those COTS D2D radios with our proposed single-die 802.11p radios integrated within phones.

3.4.1 RoadRunner Evaluation

RoadRunner [106] is an in-vehicle Android app for road congestion control, and speaks turn-by-turn navigation instructions to the driver, like existing navigation systems, while enforcing road-space rationing by allocating tokens among vehicles in the back-

ground. Tokens permit a vehicle to drive on a specific road segment, and are distributed to vehicles from a server over the cellular network (LTE), or exchanged directly between vehicles over 802.11p DSRC.

Original deployment. The original deployment took place in Cambridge, MA, USA, consisting of 10 vehicles driving among multiple possible congestion-controlled routes. Three different scenarios were evaluated: RoadRunner using only the cellular network as a baseline; additionally using ad-hoc WiFi for V2V communications; and additionally using 802.11p DSRC for V2V communications. With 802.11p, each smartphone was tethered via USB to an off-the-shelf 802.11p DSRC radio [107]. Using 802.11p enabled network response time improvements of up to 80% versus the cellular network, and cellular network usage reductions of up to 84%. Ad-hoc WiFi's performance did not suffice: with ad-hoc WiFi, only 5 V2V communications sessions occurred at an average distance of 29.2 meters, resulting in only 6.8% of requests being offloaded to V2V from the cellular network, while with 802.11p, 47 V2V sessions occurred at an average distance of 175.7 meters, offloading 43% of requests. This original deployment thus motivates the use of 802.11p as a mobile D2D communication standard for phones, while the cumbersome setup tethering a COTS 802.11p radio to a phone motivates a single-die 802.11p chip.

Adaptive power control. We obtained the RoadRunner traces and assume that with our adaptive power control, each V2V communications session (a token exchange) would be transmitted at the minimum power required to reach the other vehicle. We compare this to the original deployment traces as a baseline, in which every V2V token exchange is conducted at full radio power. The traces include vehicle location, communications on all radio interfaces, and distances at which V2V token exchanges occurred during the deployment. For each V2V exchange, we look up the minimum power level to transmit a packet across that distance from our experimental measurements of the GaN PA, using 64-QAM coding. We normalize the sum of these estimates to a situation with no adaptive power control, shown in Figure 3-5.

With ALAPC and our new PA design (22.5% efficient for all power levels), the V2V exchanges use 47% less power (from 3.3 7W down to 1.77 W), indicating that many

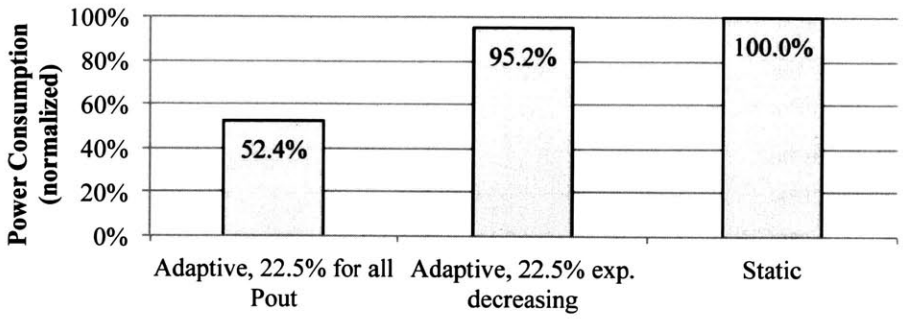


Figure 3-5: Average power consumption of RoadRunner V2V exchanges with and without adaptive power control.

V2V communications sessions did not need the full transmit power in the original deployment to reach the other vehicle. With ALAPC, but without our new PA design (so efficiency is exponentially decreasing), V2V token exchanges use 4.8% less power than the baseline (from 3.37 W down to 3.21 W), underscoring the importance of the improved PA efficiency of our circuits in realizing gains from ALAPC.

3.4.2 SignalGuru Evaluation

SignalGuru [58] is a vehicular traffic light detection iPhone app that shares data among multiple phones to collaboratively learn traffic signal transition patterns and provide GLOSA (Green Light Optimal Speed Advisory) to drivers. Each vehicle contains a windshield-mounted iPhone that observes traffic signal transitions via the phone's camera and broadcasts the observations over ad-hoc WiFi every 2 seconds.

Original deployment. The original SignalGuru deployment also occurred in Cambridge, MA, USA, along three consecutive intersections on Massachusetts Avenue. 5 vehicles followed a route for 3 hours, generating GPS location traces. To surmount the limited range of ad-hoc WiFi, a phone stationed near an intersection acted as a relay.

Adaptive power control. We obtained the SignalGuru traces, and in the simulation of our proposed 802.11p radio's performance, whenever a vehicle broadcasts a packet (every 2 seconds), we calculate the power level required to reach the nearest vehicle to it, from 19.8 to 28.8 dBm. We compare this to baseline static power control,

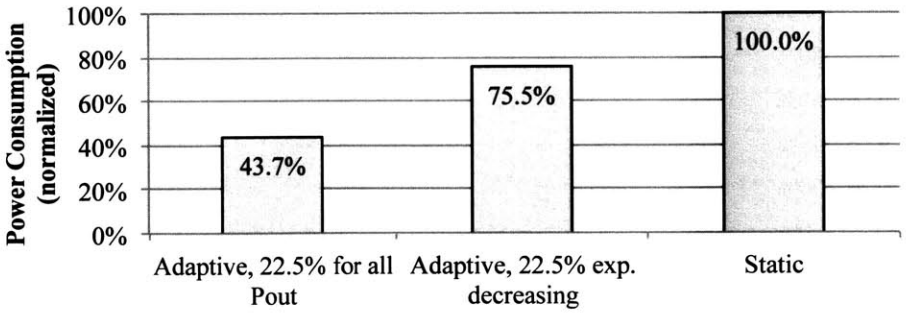


Figure 3-6: Average power consumption of SignalGuru broadcasts with and without adaptive power control.

in which every broadcast is transmitted at the maximum power level of 28.8 dBm.

With ALAPC and our new PA design (22.5% efficient for all power levels), SignalGuru broadcasts use 56.3% less power (from 3.37 W down to 1.47 W), shown in Figure 3-6. With ALAPC, but without our new PA design (efficiency exponentially decreasing), SignalGuru broadcasts use 24.5% less power than the baseline (from 3.37 W down to 2.54 W), highlighting again that the improved power efficiency of our single-die circuits is important to substantially lowering overall system power consumption.

3.4.3 Power Reduction Summary

To put our power reductions of 1.6 W (RoadRunner) and 1.9 W (SignalGuru) in context, we measured the dynamic range of a Samsung Galaxy S4 smartphone's power consumption to be between 1 W (screen on, idle) and 11 W (running a CPU-intensive benchmark) using a Monsoon Power Monitor [108]. This indicates a significant power reduction in the overall platform power budget can be realized with our new power amplifier.

3.5 Summary

This work is the result of collaboration between materials and device researchers, circuits designers and mobile systems and software architects. We leveraged a new GaN

process technology to realize the high-power power amplifier necessary for 802.11p specifications, and coupled that with a CMOS transmitter. The RF front-end circuits were tailored for adaptive power control, targeting good power efficiency across a wide range of transmit power. An 802.11p baseband processor was emulated on an FPGA (using an existing 802.11a baseband) to connect an Android phone to the RF front-end, creating a full system prototype to demonstrate the feasibility of incorporating the RF front-end into a phone. Our results show that an 802.11p front-end can be realized within the stringent power and area budgets of a smartphone to enable device-to-device communications.

Chapter 4

Device-centric communications: Enabling pervasive traffic management

This chapter contains work presented at the International Transport Systems World Congress in September of 2014, titled "RoadRunner: Infrastructure-less Vehicular Congestion Control".

This chapter presents a motivating application for next-generation ITS that leverages device-centric communications: pervasive traffic congestion management at high granularity and city-scale. The design leverages mobile sensing, processing, and D2D communications to eliminate the need for physical traffic management infrastructure, and reduce reliance on cellular communications. We build and evaluate a full end-to-end system prototype of a motivating ITS application: traffic management. We also simulate the system in a large-scale, high granularity urban simulator to measure its impact in reducing traffic congestion.

4.1 Introduction

Traffic congestion is a widespread problem affecting road transportation infrastructure in many cities, and is expected to increase in severity. In 2005, congestion resulted

in 4.2 billion hours of travel delay and 2.9 billion gallons of wasted fuel in the United States [109]. One widely studied approach to reducing congestion is road pricing, a monetary policy to disincentivize drivers from entering tolled regions. Road pricing has traditionally been implemented through manned toll booths but electronic toll collection systems are now common [110].

The Electronic Road Pricing (ERP) system deployed in Singapore in 1998 was the first in the world to apply electronic road pricing for congestion control of a large downtown area. It uses dedicated short-range radio communications (DSRC) to detect and collect tolls from vehicles passing under physical gantries on roads leading to heavily congested areas. Prices change throughout the course of a day [111, 112, 113]. In the United States, several open road high-speed tolling systems have been deployed: FasTrak in California (1993), SunPass in Florida (1999), and the Northeast's E-ZPass (1991) [114]. These systems use windshield-mounted radio transponders to communicate with physical gantries as vehicles drive by.

Congestion can also be controlled through regulatory or non-monetary policies that directly limit the number of vehicles that may drive on a road, known as road-space rationing. Prior studies in transportation research [115, 116] have shown that road-space rationing can improve road capacity when used solely or in conjunction with road pricing. However, deployments of road-space rationing around the world [117] are mostly manual, policy-driven implementations.

In Singapore, the Area Licensing Scheme [118], predecessor to ERP, required vehicles to purchase and display a paper license before entering a restricted zone (RZ). There was a fixed quota on licenses and ALS was manually enforced by officers at RZ boundaries. Now, a quota on the total number of cars in Singapore is enforced through the Certificate of Entitlement (COE) system which requires a COE to be purchased before a car can be driven in Singapore. The COE lasts for 10 years and is auction priced, with an average price of S$70K-90K in December 2012. Beijing implemented a temporary road-space rationing scheme by restricting even and odd license plate numbers on alternate days for three months prior to the 2008 Olympic Games. A slightly modified policy was implemented permanently following the suc-

cessful three-month trial [119]. London similarly enforced road-space rationing for the 2012 Olympic Games.

All above-mentioned systems require the deployment of costly physical roadside infrastructure such as gantries, tollbooths or enforcement stations and personnel, and/or specialized in-vehicle devices. Thus, deployment of congestion control tends to be limited to a few large controlled regions within cities, each covering a wide swath of roads, making it very costly to re-define regions.

A congestion control system that does not require the setup of new physical infrastructure can address these downsides of existing systems, and enable widespread deployment of congestion control across entire cities, at the fine granularity of specific roads, permitting flexible definition of regions and quotas for more responsive policies. In fact, Singapore recently released a call to companies for proposing systems for the next-generation ERP that is to be GPS-based [120], with field trials currently underway.

In this chapter, we propose, implement, and evaluate RoadRunner, an infrastructure-less congestion control system, with our prototype running on Android smartphones. Smartphones are widely adopted in many cities, with penetration reaching 50.4% [121] and 70% [122] in the U.S. and Singapore respectively. Phones can be readily plugged into vehicles, with car manufacturers providing smartphone docks on the dashboard [123], enabling seamless connectivity to substantial energy, driver-friendly interfaces, vehicle information, and vehicular communications such as DSRC [124, 125].

In the near future, RoadRunner could also be deployed on every vehicle via in-vehicle units (IVUs) rather than smartphones. All vehicles in Singapore are required to install an IVU equipped with DSRC for communications with ERP gantries, and the next generation of IVUs being field-tested in Singapore (ERP), Germany (simTD), and France (scoreF) will include GPS and 802.11p DSRC radios [35], enabling pervasive V2V communications, sensing, and computation.

Using smartphones enables an already widespread infrastructure-less solution to congestion control, but presents additional challenges. First, as smartphones and other mobile connected devices continue to proliferate, demand for cellular bandwidth

is expected to exceed available capacity by 2014 [126]. The increased throttling and cost of 3G/4G data plans and phasing out of unlimited data plans are clear symptoms of increasing bandwidth pressure on mobile data networks [38]. A phone-based infrastructure-less system permits the extension of congestion control to all roads across an entire city, but a conventional client-server implementation will lead to millions of vehicles communicating through the cellular network to servers running and policing congestion control, creating intense bandwidth pressure on already overloaded cellular networks. Second, phone-based congestion control needs to swiftly respond to drivers so they can adapt their routing appropriately. A conventional client-server implementation which relies on the cellular data network may experience long and unpredictable latencies, especially when the network is heavily loaded in dense areas [127, 128], and face difficulties meeting the real-time requirements of congestion control.

RoadRunner tackles the above challenges with a *distributed* congestion control system that offloads computing to nearby in-vehicle phones, leveraging vehicle-to-vehicle networking via ad-hoc WiFi and DSRC to ease the cellular bandwidth pressure and improve real-time response latencies. RoadRunner is a decentralized mobile phone app for vehicles to reserve places on roads in a transportation network. The system distributes tokens to vehicles as permission for their entry into regions or roads, and records infractions and/or enforces fines for violations of congestion control policies. Our prototype provides the driver with turn-by-turn voice directions just as in existing satellite navigation systems; A driver only has to enter a destination upon starting a trip, and RoadRunner automatically negotiates tokens and routing in the background. If a token cannot be obtained, RoadRunner notifies the driver of a change in route one intersection prior to the new route branch.

Our deployments on 10 vehicles show that RoadRunner improves mean system response times by 80% when coupled with DSRC radios for V2V communications, and reduces cellular data accesses by 84% compared to a traditional client-server implementation that only utilizes the cellular network. Our experiments also show that today's smartphone GPS receivers have sufficient accuracy, that, when combined

with buffer zones between regions, enable accurate identification of controlled regions entries and exits. Our simulation results (Section 4.5) show that RoadRunner can enable infrastructure-less congestion control on a large scale and improve travel speed over an existing sophisticated electronic road pricing scheme that varies tolls at different times of the day. By forgoing a charge, RoadRunner lowers costs for drivers compared to road pricing, making congestion control policies more palatable to the general public, permitting widespread deployment.

4.2 Design

At a high-level, the goal of congestion control is to ensure that there are not too many vehicles on a particular segment of road at any one time. RoadRunner is an electronic token-based reservation system: vehicles must possess a corresponding *token* to drive on a specific road segment (*token*), analogous to road-space rationing.

Regions and a quota of tokens, provided by a central server, are pre-defined by the transportation authorities. Vehicles may not create or duplicate tokens, ensuring an upper bound on the number of vehicles in a region. Tokens can expire, which helps ensure that lost tokens are effectively reset and do not impede the operation of the system over a long period. If a vehicle in the region does not have a valid token, the system logs a violation and enforces a penalty, which could be a fine or reported infraction.

When a driver steps into a vehicle and begins a trip, RoadRunner determines the route to the destination and presents turn-by-turn instructions to the driver via text-to-speech, like existing navigation systems. In the background, RoadRunner also determines whether any regions it will traverse are congestion-controlled and attempts to obtain the corresponding tokens from the server via the cellular connection or from other vehicles via the V2V radio. The pseudocode for the overall logic of RoadRunner is shown in Algorithm 1.

RoadRunner leverages V2V communications to pass tokens directly between cars when possible, using the protocol described in Algorithm 2. Each vehicle broadcasts

```
if starting trip then
    determine route to destination;
    determine which tokens need to be
      obtained;
    request tokens from server over cellular
      connection;
    if not all necessary tokens are obtained
      then
        add unfulfilled requests to
          tokensWanted queue;
        retry periodically to server;
    end
    speak prompt to driver to begin driving;
end
if nearing region then
    if token for the upcoming region has not
      been obtained then
        make a final retry to the server for the
          token;
        if successful then
            continue with no changes to the
              route;
        else
            reroute to avoid region;
        end
    end
end
if entering a region then
    if the necessary token has been obtained
      then
        mark the token as in-use;
    else
        log infraction and enforce penalty;
        create short-lived PENALTY token for
          this region;
        (PENALTY token life is 10 minutes in
          our deployment);
    end
end
if exiting a region then
    if in-use token is a PENALTY token then
        destroy PENALTY token;
    else
        remove in-use designation from token;
        place token in a tokensOffered
          collection;
    end
end
```

Algorithm 1: Overall pseudocode

```
if received ANNOUNCE from another vehicle
  then
    foreach token in (tokensWanted) do
        if token is offered in ANNOUNCE
          message then
            send other vehicle
              TOKEN_REQUEST message
              for token;
        end
    end
end
if received TOKEN_REQUEST from another
  vehicle then
    if token requested is in our tokensOffered
      collection then
        remove the token from tokensOffered;
        send it in a TOKEN_SEND message
          to the other vehicle;
    else
        ignore this TOKEN_REQUEST;
    end
end
if received TOKEN_SEND from another vehicle
  then
    if we already possess a token for that
      region then
        place this extra token into the
          tokensOffered collection;
    else
        remove the now-fulfilled request from
          the retry queue tokensWanted;
        store the token for later use;
    end
end
```

Algorithm 2: Token exchange pseudocode.

Figure 4-1: Pseudocode for RoadRunner.

an *ANNOUNCE* message every 2 seconds, containing the vehicle's ID, location, speed, bearing, and region IDs of tokens currently offered by the vehicle. *ANNOUNCE* messages are not rebroadcast or flooded because beyond 1-hop, the total latency for a token exchange exceeds V2Cloud latency: With a 40-millisecond latency for a 1-way V2V message, a 2-hop token exchange incurs 4 V2V messages with a total latency of 160 ms versus V2Cloud's 140 ms latency.

This *ANNOUNCE*, *TOKEN_REQUEST*, *TOKEN_SEND* hand-shake is necessary to ensure that each token is a singleton; if the tokens were grabbed directly from the *ANNOUNCE* message, multiple copies of the token may appear since multiple vehicles may hear the *ANNOUNCE* message. The *TOKEN_SEND* message includes the unique ID of the vehicle that is allowed to receive and use the token, so no duplicates occur among multiple vehicles. *TOKEN_REQUEST* messages arriving from multiple vehicles are processed in the order received, first-come first-served. Each *TOKEN_SEND* message is sent 3 times to minimize the possibility of token loss, and each *TOKEN_SEND* message includes a per-vehicle nonce so that extraneous receptions of the same *TOKEN_SEND* messages can be discarded. Tokens may still be lost over time, however, so we periodically expire all tokens and generate fresh tokens, in what we call **token roll-over** which occurs every hour: All the tokens in the system expire, and the server generates new tokens. Vehicles that possess an expired token can still use it, but will delete it upon usage (entering and traversing the corresponding region). This process has two purposes: 1) to ensure that any tokens lost during a V2V exchange are eventually re-injected into the system, and 2) to enable authorities to adjust the number of tokens while the system is operating, which would be difficult if it were unknown how many tokens remained in the system and might be reused indefinitely.

4.3 Implementation

We implemented RoadRunner as an Android application on Samsung Galaxy Note smartphones. The application consists of an Android Service (RoadRunnerService)

that implements the main logic of RoadRunner and continuously runs in the background, and an Android Activity (MainActivity) that shows the status of the application. RoadRunnerService and MainActivity run in the same thread. A separate thread manages the V2V communications interface (Wi-Fi or DSRC), running a busy-wait loop to receive packets from the network interface associated with the V2V radio. We were not concerned with energy consumption as the smartphone can be plugged into the vehicle's power source.

The smartphones communicate with the remote server over 4G LTE cellular data, which represents the state-of-the-art in mobile data access today. On the server, we implemented a Python application that services requests over TCP through a line-based protocol, allowing vehicles to make requests (a GET request) for and receive tokens from the server, and to send tokens back to the server (a PUT request). If there are no tokens available for a requested region, the server will respond with an error code (GET 500 FULL).

For V2V communications, the app can leverage either 802.11p DSRC or 802.11n adhoc Wi-Fi. We implemented support for both interfaces in our Android application, and include 802.11n in our evaluation to demonstrate that WiFi's range limitations render it inadequate for V2V communications.

To use 802.11p DSRC, we connect the Androïd smartphone to a Cohda Wireless MK2 WAVE-DSRC Radio [107] through the MK2's USB 2.0 OTG host interface. We enable USB tethering on the smartphone, which enumerates the phone as a generic USB CDC Ethernet interface on the MK2 host. FwdWsm, a software bridge application on the MK2, receives UDP packets from the phone on the ethernet interface, encapsulates them in WAVE Short Message (WSM) packets, and broadcasts them over the 802.11p wireless interface. FwdWsm also receives WSM packets, removes the WSM headers, and forwards the resulting UDP packets to the USB Ethernet interface. Thus, our Android app can communicate with the 802.11p radio by sending and receiving UDP packets. The MK2 radios utilize dual roof-mounted 5.9 Ghz antennas, and are powered by the 12V power supply from the vehicle's cigarette lighter port.

To use 802.11n adhoc Wi-Fi, we run Android 2.3 on the Galaxy Note smartphones as we require support for wireless extensions (WEXT), available only in the Android 2.3 drivers for the Broadcom BCM4330 chipset. We cross-compiled *iwconfig* to configure the cards into adhoc mode at the lowest bitrate supported (1Mbps) with power management disabled. Each smartphone is configured with a unique static IPv4 address, and all V2V communications happens over UDP broadcast.

4.3.1 Practical considerations

Electronic tolling/road-pricing. While RoadRunner is implemented and deployed as an area-space rationing system, it can be readily modified to support conventional road pricing. A time-based road pricing scheme, such as that used in Singapore's ERP [111], where a different rate is charged at different times, can be straightforwardly implemented with RoadRunner by having the server attach prices to tokens at the start of a time interval when first distributing, and having tokens expire at the end of a time interval. Every time a vehicle receives a token, it deducts the corresponding value from the vehicle's account. Thus, every time the tokens expire, a new batch of tokens with new prices can be generated on the server.

Cars already in the region can contact the server a short time (randomized to spread out V2Cloud load) before the deadline and trade in their soon-to-expire tokens for fresh ones, while any tokens not traded in would be considered lost and freshly regenerated by the server. While this would require every car holding tokens to contact the server, RoadRunner would still have most of its beneficial effects on the cellular network as token exchanges that occur outside of token renewals can still be offloaded to V2V. Transportation authorities can tune how long-lived tokens are to trade off between more frequent V2V offload and more frequent token renewals and pricing updates.

The payment account can be implemented via a prepaid smartcard inserted into an interface to the smartphone, as is currently done with a specialized in-vehicle unit in Singapore's ERP [111] system, or it can be electronically managed via online accounting systems like PayPal.

Vehicles origin or destination within a region. If a vehicle ends a trip or parks in a region that it currently has a token for, it can return the token to the cloud server. If a vehicle begins a trip in a congestion-controlled region, transportation authorities can choose how to enforce policy: 1. vehicles originating in a region may not begin moving (detected from GPS speed) until RoadRunner acquires a token for the region, or 2. vehicles originating in a region do not count towards that region's quota.

Failure of cloud server and/or cellular network. In the event of a cloud server or cellular network failure, the system can still operate purely over V2V, as vehicles will indefinitely offer tokens over V2V if they cannot return them to the cloud server, and other vehicles can still request those tokens over V2V communications. Since vehicles are no longer able to exchange tokens via V2Cloud communications (GET from and PUT to cloud), there may be more unnecessary PENALTY infractions as some vehicles may never come within V2V range of desired tokens. Vehicles which end their trips out of V2V range will cause the tokens they are holding to be lost, which will cause a decrease in the total number of tokens in the system over time. Thus RoadRunner can survive V2Cloud communications outages, but only for a limited duration.

4.4 Deployment of RoadRunner Prototype

We evaluated RoadRunner by comparing the performance of RoadRunner with V2V communications and token exchanges enabled to a Cloud-only baseline. The Cloud-only variant communicates solely with a remote server via an LTE cellular connection, and the V2V-enabled variant additionally communicates with other vehicles over 802.11p DSRC. To demonstrate the inadequacy of adhoc Wi-Fi for V2V communications, we also evaluated a variant that used adhoc WiFi instead of 802.11p DSRC. The server portion was implemented as a Python application that serviced requests over HTTP, and was located in the same geographic region as the phones to minimize backbone Internet latency.

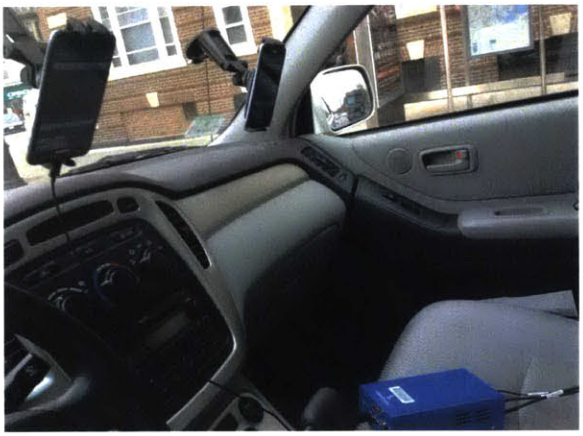


Figure 4-2: Setup in each vehicle.

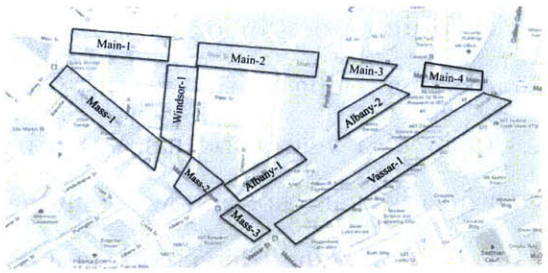


Figure 4-3: Map of deployment regions.

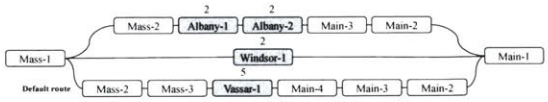


Figure 4-4: Deployment routes. Controlled regions have capacity shown.

Our deployment took place in eastern Cambridge, MA, USA (Figure 4-3), a triangular region with a base width of 775 meters and a height of 315 meters. We split each road into regions between intersections which served as the buffer zones, resulting in a total of 11 regions on 2570 meters of road, with 4 controlled regions having a bounded number of tokens available for each, and 7 unrestricted regions that did not require tokens to traverse (Figure 4-4). This resulted in a fine-grained congestion control scenario of 4 controlled regions with a total distance of 900 meters, whereas Singapore's Orchard Road ERP zone is 1 controlled region of 2200 meters, an order of magnitude coarser.

Ten vehicles participated in our experiment, driving along a default loop through Mass. Ave, Main St and Vassar St, with half of the vehicles going clockwise, and the other half going anti-clockwise. The RoadRunner app provided voice-over instructions to drivers to divert to Windsor St or Albany St depending on the success/failure in obtaining the necessary tokens. Vehicles circulated among the regions for 20 minutes beforehand to reach a random steady-state distribution of vehicles over the deployment area. Each vehicle had two smartphones mounted on the windshield, one connected to a DSRC radio and the other utilizing its internal WiFi radio as in Figure 4-2.

We selected token quotas for each region to ensure that at least some of the ten vehicles had to divert onto alternative routes because they were unable to obtain a token, and to ensure that some vehicles would be unable to obtain tokens for any route and would incur a PENALTY reservation. Due to RoadRunner's design, if a vehicle is in a congestion-controlled region or in an adjacent region, it will either have a token in-use or have an outstanding token request, except in the single intersection between Main-2 and Main-3 which is not adjacent to any congestion-controlled regions. Thus, our selection of token quotas and controlled regions in Figure 4-3 resulted in all the major scenarios: obtaining all necessary tokens for the default route, not obtaining tokens for the default route but obtaining necessary tokens for alternative route(s) alternative routes, and not obtaining necessary tokens for any route and thus incurring PENALTY reservations.

For the V2V-enabled variants, we tested the two possible operation paradigms of RoadRunner: 1) an on-demand navigation and routing system that requests tokens just-in-time for the next region, and 2) a pre-reserve system that requests all necessary tokens at the beginning of each trip iteration. For the Cloud-only variation, we did not test on-demand vs pre-reserve because they are effectively the same: all requests end up going through the remote server anyway, and unused tokens do not remain on the vehicles for V2V exchanges. Only the cloud server has any available tokens, so it does not matter whether vehicles check with the cloud at the beginning of a trip or on-demand.

We ran two ten-minute trials each of V2V on-demand over WiFi, V2V on-demand over DSRC, V2V pre-reserve over WiFi, V2V pre-reserve over DSRC, and Cloud-only.

On-demand requests versus pre-reserve requests. RoadRunner can operate in two system-wide modes for V2Cloud communications: Pre-reserve and On-demand. In both modes, vehicles watch for and attempt to obtain desired tokens over V2V communications at all times. These two modes represent a trade-off between providing more certainty about the route a driver will take at the beginning of a trip vs. improving token utilization, which directly impacts road usage.

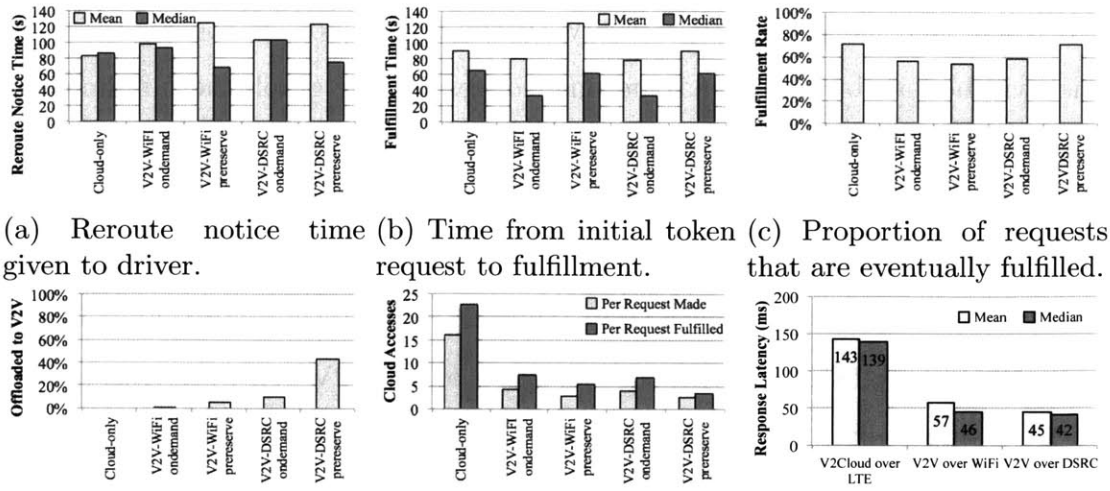
Pre-reserve requests mode allows RoadRunner to make token requests for all the

tokens it needs for a route at the beginning of a trip over V2Cloud. The vehicle periodically reattempts any remaining unfulfilled token requests, too, increasing the frequency of V2Cloud token requests in the system. Furthermore, vehicles hold tokens for a longer period of time without actively using them, decreasing token utilization rates. Pre-reserve requests may provide a better user experience, however, since the system can show the driver a complete, preferable route at the beginning of the trip if available, rather than as a reroute while the user is already driving.

On-demand requests mode defers the V2Cloud token request for each region until the vehicle arrives at the intersection immediately before entering the region, reducing the frequency of V2Cloud communications and reducing the time that tokens may remain unused on a vehicle before it has reached the region. This may provide a poorer user experience, however: more preferable routes may become available only during the drive, requiring reroutes and imposing uncertainty.

Server retry timeout. Unfulfilled token requests are periodically retried to the server, determined empirically: 2 seconds for the cloud-only variant because tokens can only come from the server, 10 seconds for the V2V-enabled on-demand variant because tokens may come from other vehicles, and 30 seconds for the V2V-enabled pre-reserve variant because tokens are even less likely to be on the server since tokens are requested at the beginning of a trip.

Token PUT timeout. When a vehicle exits a region, it waits before returning the token to the server. A longer timeout increases the window for a V2V token exchange, but a shorter timeout decreases the duration a token may sit unused on the vehicle. We determined timeouts empirically: 10 seconds for the V2V-enabled on-demand variant because there are fewer unfulfilled requests in the system overall as requests are made on-demand when nearing a region, and 60 seconds for the V2V-enabled pre-reserve variant because there are more outstanding unfulfilled requests in the system overall as tokens are requested at the beginning of the trip.



(a) Reroute notice time given to driver.

(b) Time from initial token request to fulfillment.

(c) Proportion of requests that are eventually fulfilled.

(d) Proportion of requests that are fulfilled over V2V.

(e) Average num of cloud accesses over cell network.

(f) System end-to-end request-response latency.

Figure 4-5: RoadRunner deployment measurements

4.4.1 Request fulfillment offload

Figure 4-5d shows the proportion of all fulfilled token requests that are fulfilled over V2V (in the Cloud-only variant, all requests are fulfilled over V2C). V2V-WiFi is not able to offload many token exchanges, due to the limited range of Wi-Fi: in the deployment, only 5 token exchanges occur over WiFi at a mean distance of 29.2 meters, while 47 token exchanges occur over DSRC under the same conditions, at a mean distance of 175.7 meters. V2V-DSRC is able to offload a significant portion of token exchanges, up to 43%. The pre-reserve variants offload more than the on-demand ones as requests for each region are made at the beginning of the trip rather than just before the region, giving vehicles more time to encounter a token offered over V2V.

4.4.2 Request fulfillment time

All token requests are timestamped when the request is created and when the request is finally fulfilled, whether by the centralized server (V2C) or by another vehicle (V2V). If the quota for a region is too low, fulfillment times may be unnecessarily long, and result in sub-optimal road throughput as cars are throttled waiting for tokens

even when there is spare road capacity. If the token quota is too high, fulfillment times will be low, but roads may become congested. We graph the fulfillment times for the variants of RoadRunner in Figure 4-5b.

With good V2V communications, the RoadRunner distributed token reservation protocol is able to match the request fulfillment times of the Cloud-only baseline. Average fulfillment times for both DSRC and WiFi variants of RoadRunner are similar to the Cloud-only baseline, with the exception of the V2V-WiFi pre-reserve variant, which obtained only 5 tokens over V2V out of 73 total tokens obtained (a 6.8% ratio) due to the limited range of WiFi. In contrast, V2V-DSRC pre-reserve obtained 37 tokens over V2V out of 86 total, or 43% of tokens. DSRC's improved range allowed it to more often bypass the wait for a token to appear on the cloud, reducing the fulfillment time. On-demand variants have much lower median fulfillment times than cloud-only and pre-reserve variants because requests for a region are made just-in-time in the prior region.

4.4.3 Request fulfillment rate

The fulfillment rate is the proportion of token requests that eventually do acquire a token. The fulfillment rate is useful for understanding the effects of the more unreliable vehicle-to-vehicle communications and token exchange protocol vs. the reliable cellular and cloud server connection (available 91% of the time in our deployment). (Note that a fulfillment rate of 100% is undesirable since it is the same as no congestion control: every vehicle can enter a controlled region.) We use the V2C fulfillment rate as a baseline, and expect that V2V RoadRunner should result in similar fulfillment rates. Any significant deviations would imply that the V2V variants are negatively impacting the ability of vehicles to obtain tokens, beyond the effects of traffic congestion.

We show the fulfillment rates for the variations of RoadRunner in Figure 4-5c. DSRC RoadRunner show similar fulfillment rates to the Cloud-Only baseline, implying that when using DSRC for V2V communications, the distributed road reservation protocol of RoadRunner successfully fulfills token requests just as well as a Cloud-only

implementation.

The WiFi variants show poorer fulfillment rate, indicating that the distributed road reservation protocol is negatively impacting the ability of vehicles to obtain tokens, due to WiFi not having sufficient range to meet other vehicles with tokens. Pre-reserve DSRC Roadrunner has better fulfillment rates than on-demand DSRC Roadrunner as token requests are created at the beginning of a trip rather than on-demand, giving the vehicle more time to encounter a nearby vehicle offering that token: indeed, DSRC pre-reserve was able to fulfill request over V2V more frequently (43.0% of all requests vs. 10.6% for DSRC on-demand).

4.4.4 Reroute notice time

The reroute notice time is the time from when the route changes (a reroute) due to a token being newly acquired, to when the driver turns onto the new route. Reroutes occur if a more preferable route becomes available; when this happens, we automatically update the navigation route to the most preferable, present updated turn-by-turn voice navigation directions to the driver, and display tokens in possession on the screen. If the driver takes a different route or is unable to turn onto the new route in time, RoadRunner offers the tokens over V2V or returns them to the server.

In our deployment, the route passing through Windsor-1 is the shortest and most preferable, the route passing through Albany-1 and Albany-2 is the next most preferable, and the route through Vassar-1 is the least preferred. The Vassar-1 route is the default route presented to the driver, and if no tokens are available, the driver will incur a *PENALTY*.

The reroute notice times for the variants of RoadRunner are shown in Figure 4-5a. In all but one case in the Cloud-only variant, drivers had at least 50 seconds to turn onto the route.

The on-demand V2V variants of RoadRunner outperformed the Cloud-only baseline in reroute time provided to the driver. The pre-reserve V2V variants had a bimodal distribution: when vehicles are able to pre-reserve tokens in advance at the beginning of the trip, this counts as a reroute away from the default, longest route

and thus those lucky drivers are afforded a large amount of time to take the new route. For drivers who did not get those tokens, however, they often get tokens just-in-time as the previous group of lucky drivers finish using their tokens and make them available to the latter group.

4.4.5 System responsiveness

We characterized the Vehicle-to-Vehicle (V2V) and Vehicle-to-Cloud (V2C) interactions in our deployment for an apples-to-apples comparison. All token exchanges are timestamped on the phones from request sent to response received to obtain end-to-end system latencies. We compare the latencies for interactions occurring Vehicle-to-Cloud over 4G LTE (V2C), Vehicle-to-Vehicle over adhoc Wi-Fi (V2V-WiFi), and Vehicle-to-Vehicle over DSRC (V2V-DSRC).

V2V latencies, shown in Figure 4-5f, are significantly lower than V2C latencies, with interactions over WiFi showing 61.2% reduction in mean latency and 22.5% reduction in median latency, and DSRC showing a 79.9% reduction in mean latency and 62% reduction in median latency. V2C latencies have a much higher mean than median due to a long-tail distribution in which some cellular accesses taking a disproportionately long time to complete. These findings are consistent with prior characterization studies [74, 127, 128].

DSRC latencies are not as low as the 100 microseconds delay requirement for safety applications or previously measured DSRC latencies [129], as we have additional delays incurred from the use of the FwdWsm software bridge, the USB Ethernet interface to the phones, the Android stack and Dalvik VM that Android apps run within, and the RoadRunner application overhead. Congestion control is not a safety application, however, and DSRC RoadRunner already shows significant improvements over the conventional client-server implementations of prior infrastructure-less electronic tolling systems [130] [131] [132] that rely solely on cellular.

4.4.6 Cloud access offload

For each of the RoadRunner variants, we measure the ability of the system to reduce the load on the cellular data network. For each variant, we divide the total number of requests made to the cloud server over the LTE connection by the number of token requests successfully fulfilled. Figure 4-5e shows that all V2V variants of RoadRunner are able to reduce cloud accesses per token significantly compared to the Cloud-only variant with reductions ranging from 66.3% to 84.3%.

These results demonstrate the benefits of leveraging V2V communications to exchange tokens over a conventional client-server implementation. We achieved reductions up to 84% in cloud accesses incurred per request, and latency reductions up to 80%. The RoadRunner distributed token protocol running over DSRC matches the fulfillment rate of a Cloud-only baseline and does not significantly increase unnecessary penalties on controlled regions.

Due to our limited deployment size of 10 vehicles, it is difficult to measure RoadRunner's effectiveness in mitigating traffic congestion at realistic scale: the token quotas are so low (only 2-5) that infractions are enforced when the region is not even close to capacity because some tokens have been reserved by cars not yet in the region. Instead, we rely on our following simulation studies (Section 4.5) to demonstrate the enforcement and congestion control effectiveness of RoadRunner.

4.5 Large-Scale Simulation

While our deployments provided us with measurements of RoadRunner's application and network performance, it is also critical to evaluate RoadRunner's effect as a transportation policy vs. existing road pricing schemes. Policy-wise, a road-space rationing policy like RoadRunner allows drivers to enter road segments free-of-charge when the congestion level is below the quota, making it more palatable to drivers. Here, we evaluate RoadRunner's effectiveness on transportation policy metrics like travel speed and road capacity/throughput vs. existing Electronic Road Pricing (ERP) in Singapore. We used the SimMobility short-term simulator (SimMobilityST) [133],

an agent-based, multimodal microscopic simulator where drivers, pedestrians and passengers are modeled as agents whose behavior and movement are captured at a very fine resolution of milliseconds. SimMobilityST models detailed human behavior, including drivers changing lanes, accelerating/braking, choosing routes, and how pedestrians walk, how people board buses, etc.

We recreated the movement of vehicles under ERP and simulated individual driver behavior, vehicle movement, the RoadRunner app, and communications latency and range within SimMobilityST, for an existing road pricing region in Singapore, with realistic vehicle traffic generated from actual loop detector information, at 10-millisecond resolution (necessary to model network communications latencies of 50-200 ms).

We simulated the RoadRunner on-demand app with a congestion control policy providing various numbers of road reservation tokens on Orchard Road. We did not simulate RoadRunner pre-reserve as the loop count data does not provide true trip origins outside the Orchard Road area road network, which are necessary to simulate making pre-reserve requests at the beginning of a trip.

Traffic movement modeling. We start with loop detector counts from the Orchard Road ERP region in Singapore, collected over a 24 hour period beginning Thursday, August 5, 2010. Vehicle counts are available for eight intersections on the Orchard Road road pricing region (Figure 4-6). For each intersection, we manually annotated which detectors counted vehicles turning onto (entering) the region, which detectors counted vehicles turning out of (exiting) the region, and which detectors counted vehicles continuing to travel inside the region.

The raw loop count data provides the number of vehicles crossing through the intersection of each lane every 5 minutes, detected by loop detectors embedded under each lane. To simulate the movement of individual vehicles within the region, we create Origin-Destination (OD) pairs that represent vehicle trips. We assume a Poisson process for vehicle detections, and thus distribute detections randomly across the 5 minute time interval into 10 millisecond bins.

For each of the roads crossing Orchard Road, we select an Origin point on the

crossroad 250 meters before the intersection, for each possible direction that a vehicle can enter from. Similarly, we select a Destination 250 meters after the intersection for each possible exit direction. Each loop detector is paired with the corresponding Origin and Destination point.

For each detected exit, we correlate it to a Destination point based on which intersection and lane the loop detector is in, and pair it with one of the valid Origins from which it could have come from, sampled from the distribution of entrance loop count detections in the same time-step. We assume that the distribution of cars across the entry points varies slowly relative to the travel time of a single vehicle within the 2.2 km long road. This resulted in 74,904 Origin-Destination pairs representing trips through Orchard Road.

4.5.1 Simulation setup and parameters

We simulated the three variants of RoadRunner On-demand (Cloud-only, V2V-Enabled with DSRC, V2V-Enabled with WiFi) and compared it to an unmodified simulation of the baseline ERP policy in which all the Origin-Destination pairs travel through Orchard Road.

With the RoadRunner variants, if RoadRunner cannot obtain a token, it finds an alternate route that avoids Orchard Road and reaches the Destination point via other roads near Orchard Road and reroutes the driver. If it cannot find an alternate route, the driver continues onto Orchard Road but incurs a penalty reservation, similarly to the real deployment.

We simulate a V2V communications range and latency of 175.7 meters and 45 milliseconds for DSRC and 29.2 meters and 57 milliseconds for WiFi, with a 100% message reception rate, based on the average token exchange distances and latencies in our real deployment (Section 4.4). We also simulate a cloud server that exchanges tokens with the vehicles over a V2Cloud cellular connection, with 143 ms latency and 100% availability. (The cellular data connection was available 91% of the time in our deployment, and 100% with retries).

We initialize the simulation with no vehicles within the region when the simulation

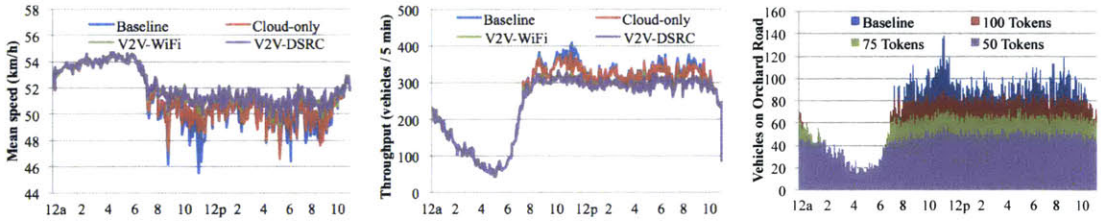
Figure 4-6: Map of Orchard Road and intersections for simulation.

begins at 12:00am. Over the course of the simulation, for the baseline ERP policy, this builds up to a maximum of 139 vehicles on Orchard Road at 11:37am.

At each time-step, SimMobility simulates driver behavior (navigation, lane changes, maintaining following distances, etc.) and the RoadRunner app, which operates in the same manner as the RoadRunner app in the deployment, with a few differences:

Localization. SimMobility provides the vehicle location directly to the app.

On-demand V2Cloud requests. As the vehicle travels along a crossroad, RoadRunner requests a token on-demand when the vehicle is 50-70 meters away from the intersection with Orchard Road (using a 100x100 meter bounding box)



(a) Mean speed, 100 tokens. (b) Throughput, 100 tokens. (c) Occupancy, V2V-DSRC.

Figure 4-7: RoadRunner Orchard Road simulation measurements

4.5.2 Simulation results

Quota enforcement: Figure 4-7c compares the vehicular occupancy of Orchard Road for the Cloud-only, V2V-DSRC, and V2V-WiFi variants when 100 tokens are allocated. RoadRunner successfully reduces the number of vehicles in the region

according to the number of tokens, even when some drivers cannot find an alternate route and thus must traverse Orchard Road anyway.

Travel speed and throughput: Figure 4-7a compares the average speed of vehicles with different token allocations for the V2V-DSRC variant. We sample the speed of every vehicle on Orchard Road once per second and aggregate the samples into 5-minute bins. RoadRunner improves the travel speed of vehicles at times of congestion compared to the ERP baseline. It significantly eliminates many peak congestion periods and improves minimum travel speeds: the slowest average travel speed across the 24-hour period improves 7.7% from 45.5 km/h in the ERP baseline to 49.0 km/h with V2V-DSRC. Overall vehicle throughput is reduced during peak congestion in our evaluation: some tokens are not circulated immediately to vehicles demanding them, and in other cases, improving travel speed necessarily reduces throughput. The number of tokens can be adjusted by transportation policy planners to trade-off between vehicle speed and total throughput.

Cellular data access reduction: In the simulations with 100 tokens allocated, the Cloud-only baseline recorded 73,693 cellular network accesses to GET or PUT tokens. The V2V-WiFi variant reduced the number of these cellular accesses by 14.5%, and the V2V-DSRC variant reduced cellular accesses by 24.8%. This reduction is lower than in the deployments because there is only one region, so no additional token exchanges occur between vehicles for different regions.

4.6 Related Work

RoadRunner is related to several previous systems in infrastructure-less congestion control, but differs by leveraging direct V2V communications instead of relying purely on a centralized client-server design using the cellular data connection. It also realizes congestion control with road-space rationing rather than conventional road pricing, a policy that we believe is much more acceptable to the general public for a pervasive, city-wide implementation. Lu et al [130] and Lee et al [131] demonstrate GPS-based tolling systems with tolling information reported to a server through a GPRS or 3G

cellular connection. Srinivasan et al [132] present a map matching and development platform for infrastructure-less electronic road pricing systems that runs on mobile devices, which can be applied to RoadRunner for more accurate localization.

RoadRunner is not a traditional vehicular network as it combines a reliable cellular connection and restricts vehicular routing to a single hop to keep response times low, but the following systems provide valuable insights on message routing, vehicular positioning, and security. Leontiadis et al [134] present a geographic routing protocol for vehicular networks and simulate using vehicle traces. Wu et al's MDDV [135] is an algorithm for data dissemination over V2V that combines opportunistic, trajectory based, and geographical forwarding, applicable to keeping tokens geographically near their regions. MaxProp [136] routes message between peers without knowing the state of a partitioned disruption-tolerant network or the meeting locations. Wisitpongphan et al [137] show that conventional routing techniques such as AODV or DSR do not work for sparse vehicular adhoc networks, such as on a RoadRunner controlled region during times of low traffic. Boukerche et al [138] examine the suitability of data fusion techniques to provide robust localization for vehicular networks, which could help improve our controlled region granularity. Parno et al [139], Raya et al [140], and Lin et al [141] contribute protocols, discussion, and designs on securing vehicular networks, critical to ensuring malicious users do not defraud or disable RoadRunner.

Chapter 5

Device-centric processing: Enabling pervasive computing for ITS

This chapter contains joint work with Anirudh Sivaraman, HaoQi Li, and Niket Agarwal, presented at the International Conference on Robotics and Automation (ICRA) September 2012, titled "DIPLOMA: Consistent and Coherent Shared Memory over Mobile Phones."[1]

This chapter presents an architecture for leveraging the increasingly powerful computational capabilities of mobile devices across a network of mobile phones to build highly-responsive ITS applications. Today, even with increasingly powerful mobile CPUs and GPUs, networked mobile apps still typically use a client-server programming model, sending all shared data queries and uploads through the cellular network, incurring bandwidth consumption and unpredictable latencies. Leveraging the compute power of modern smartphones and device-to-device communications can mitigate demand on cellular networks and improve response times. We present DIPLOMA, which aids developers in achieving this vision by providing a programming layer to

easily program a collection of smartphones connected over D2D communications. It presents a familiar shared data model to developers, and behind the scenes, it implements a distributed shared memory system that provides coherent relaxed-consistency access to data across different smartphones while addressing the issues that device mobility and unreliable networking pose against consistency and coherence.

5.1 Introduction

As mentioned in Chapter 2, pervasive mobile devices can sense and generate large amounts of data, and are increasingly able to process the sensed data in-situ as well. Yet, mobile phone applications still use the conventional client-server model, with a thin client front-end on the phone delegating compute-intensive tasks to servers in the cloud. This model is widely used for simplicity, but has several disadvantages in a mobile context:

1. **Overloading of cellular networks:** Wireless spectrum is at a premium, and the growth in demand for mobile data is outstripping new capacity even as cellular communications technologies are evolving [126, 38]. Additional demand comes as smartphone adoption continues to increase world-wide, 4G networks increasingly serve as home broadband connections, and higher screen resolutions on mobile devices are increasing user demand for high bandwidth content like streaming video.

2. **Long and variable latencies:** Cellular networks are characterized by long and highly variable latencies, degrading application response times [127, 128]. Our own measurements in Section 5.4 confirm that 3G latencies can be as high as 50 seconds, and while 4G latencies are significantly better, 4G latencies are still an order of magnitude higher than WiFi latencies.

3. **Poor battery life:** Cellular data transmission drains energy [142], a primary resource for mobile phones.

4. **Monetary cost:** Cellular service plans are increasingly metered and monthly caps are now common [38].

We propose moving to a shared memory programming model for location-based services, addressing the issues of cellular network overload by leveraging Device-to-Device (D2D) wireless communications to eliminate the need for cellular communications when possible. Application developers see a single global address space as our programming layer creates a shared memory abstraction and hides the underlying mobility and phone-to-phone coordination. We present the following contributions:

1. We design and implement DIPLOMA (Distributed Programming Layer Over Mobile Agents), enabling distributed programming by exposing a shared memory model to the application developer (Sections 5.2 and 5.3).

2. We implement an app similar to the popular location-based photo sharing service on Google Maps, Panoramio [143], and a synthetic benchmark. We measured substantial benefits in latency and cellular bandwidth reduction compared to a conventional client-server implementation on 3G and 4G (Section 5.4).

5.2 The Design and Semantics of DIPLOMA

At a high level, a collection of mobile smartphones is a distributed system with each device having a processor core and memory. Devices are interconnected by short range radios such as ad-hoc WiFi. We propose that devices cooperate and share their memory[2] to form a distributed shared memory (DSM) system to present a familiar interface to developers. However, typical DSM systems use static nodes connected over a reliable interconnect, while a collection of smartphones represents mobile nodes connected via unreliable wireless networking. To address device mobility, we divide a geographical area into a 2D mesh of regions. Within each region, we abstract the collection of all phones in the region into a *single, reliable and immobile* Virtual

[2]Mobile apps are typically sandboxed, so their effects on the system are isolated, mitigating security concerns. Additionally, future mobile virtualization can further isolate DIPLOMA apps [144].

Core (VCore) with its own memory (Section 5.2.1). To address the unreliability of the wireless interconnect, we relax our memory consistency model (Section 5.2.2). Additionally, we cache to speed up remote reads, and propose Snoopy, Resilient Cache Coherence (SRCC) to maintain coherence (Section 5.2.3).

5.2.1 The Virtual Core layer (VCore)

VCores provide the abstraction of static reliable cores interconnected via a 2D mesh. We leverage Virtual Nodes(VN) [145], which abstracts a collection of unreliable mobile nodes in direct communication range of each other[3] into a stationary reliable virtual node. In the original VN system [146], a large geographical area like a city is first divided into equal-sized regions. Mobile nodes can infer their region via localization (e.g. GPS). Region size is chosen based on radio range, such that messages sent from one region can be heard by all nodes in the region, as well as in all neighboring regions. All physical nodes in a region participate in a state replication protocol to emulate a single VN per region.

The nodes elect a leader using a simple algorithm. Each node, on entering a new region, sends a leadership request to all nodes. If the leadership request is not rejected, the node claims itself as the leader and sends out regular *heartbeat* messages announcing its leadership. If a non-leader misses a certain number of *heartbeats*, it sends out a leadership request.

The client nodes broadcast requests to their local region. The leader, and non-leaders, run the same server application code. All nodes receive client requests and process them according to the application code. Only the leader node sends responses; others buffer responses until they hear the same response message from the leader. By observing the leader's replies, the non-leaders synchronize their application state to the leader and correct themselves upon a state mismatch.

The only practically deployed implementation of VN is described in [146], on a small set of PDAs. Another implementation [147] simulates VNs on the ns-2 [148]

[3]DSMLayer, described later, removes this constraint so deployments can span arbitrarily large geographic areas.

simulator. These original VN systems run into problems in practice due to unpredictable mobility and unreliable networking. Regions could become unpopulated, causing VNs to lose state. Wireless contention and range issues can create multiple leaders if nodes do not hear *heartbeats*, causing inconsistent state.

Proposed Virtual Cores. To address these problems, we propose a new implementation called Virtual Cores (VCores). A VCore is the leader in a group of mobile nodes in a single region. Most anomalies in Virtual Nodes occur when the elected new leader is out-of-sync with the old leader. VCores correct this via occasional coordination with a reliable cloud server using cellular networks like 3G (HSPA) or 4G (LTE).

Region boot-up: When the first mobile node enters a region, it broadcasts a leadership request message. If there is a VCore running here, it replies to the request and the new node becomes a non-leader. If the new node does not hear a reply within a timeout period, it contacts the cloud to nominate itself as a leader. The cloud knows if a VCore is already running in the region, and rejects the leadership request if so. Otherwise, it sends the latest shared memory state of this region back to the node, which then boots itself as the region's new VCore.

Leader (re)election: The VCore provides a stationary, reliable core abstraction until it leaves the region. At this point, it broadcasts a *LEADER_ELECT* message back to the old region. The nodes in the old region receive this message and reply with a *LEADER_NOMINATE* message. The old VCore randomly chooses one to be the new VCore and sends it a copy of the shared state with a *LEADER_CONFIRM* message. The new VCore sends a final *LEADER_CONFIRM_ACK* message to the old VCore. If the election fails due to message losses or if the old region is unpopulated, the old VCore sends the shared state to the cloud for later retrieval by a new VCore. The above steps ensure that if the region is populated, exactly one node in this region will be selected as the new VCore.

No state replication: In the original VN, the leader's state is replicated on all non-leaders, which keep their state synchronized with the leader by observing requests and the leader's replies. We eliminate replication since it does not improve reliability:

the cloud server has to confirm leadership requests anyway to ensure consistent state.

5.2.2 The DIPLOMA Shared Memory layer (DSMLayer)

DSMLayer is implemented as an API that runs atop the immobile and static VCore abstraction which is overlaid over individual phones. DSMlayer glues VCores in a grid/mesh topology, communicating via wireless multi-hop messages between adjacent VCores. The phone currently running the VCore for a region contributes part of its memory towards the global shared memory, addressed through variable names rather than binary addresses. These variables make up the *shared address space* of DSMLayer. Each shared variable resides on one VCore, its *home* VCore. Variables are accessed consistently through the **Atom** primitive, which is a block of instructions executed atomically on the shared variables resident on a single *home* VCore. To execute an Atom, it is multi-hop forwarded[4] from the originating VCore to the *home* VCore and executed on its portion of shared memory.

Atoms are atomic, and always execute once or fail completely. They are equivalent to a critical section, or an *acquire-release* block in Release Consistency (RC) [149]. We discuss similarities and differences with RC in detail in Section 5.5. We guarantee relaxed consistency [150] by default and allow Atoms to be reordered by the unreliable wireless network. To optionally enforce stricter ordering between atoms, we provide *AtomFence*, a per-*home* VCore memory fence primitive that can be executed before an Atom to guarantee that all previous Atoms occurring in program order in the thread have completed. The use of AtomFence is optional: for some applications, allowing reordering improves performance.

Additionally, DIPLOMA provides *at-most-once* [151] execution semantics for Atoms by logging the reply when an Atom is executed. Thus, if a duplicate request is received due to a retry, the logged reply is sent back without re-execution.

[4]Beyond a certain threshold of hop count, ad-hoc WiFi energy and latency will exceed those of cellular networks, and a hybrid cloud/WiFi solution would be better

5.2.3 Snoopy and Resilient Cache Coherence (SRCC)

Accesses to remotely homed data result in round-trip (possibly multi-hop) communications between the requesting and *home* VCores; resending lost messages exacerbates these delays. Caching addresses this problem, but necessitates a coherence protocol. We explain our design choices below.

Traditionally, coherence protocols are either broadcast-based [152] or directory-based [153]. In a wireless context, the latency of an extra hop (required by directory-based protocols) is high and communication is inherently broadcast, so *broadcast*-based protocols are a better fit. Further, *write update* protocols are more suitable than *write invalidate* protocols since write update protocols result in fewer messages exchanged. They consume more bandwidth by carrying the shared data in each message, but WiFi bandwidth is sufficient. Additionally, we use a *write-through, no-write-allocate* cache to ensure writes do not appear in the local cache until the local VCore receives a write update confirming the write is complete at the remote home VCore. To ensure memory consistency, all cached copies in the system must see the same **order** of reads and writes to a particular memory address. We build on timestamp snooping [154] and INSO [155], which are multiprocessor broadcast-based protocols that achieve ordering on unordered networks by assigning ordered numbers to coherence messages and presenting them in order to the destination caches. INSO and timestamp snooping rely on a highly reliable interconnect, however, making them unsuitable for wireless networks. *DIPLOMA requires a novel write update, snoopy (broadcast-based) cache coherence protocol resilient to unreliable networking.*

We design a Snoopy and Resilient Cache Coherence (SRCC) protocol. SRCC guarantees that memory operations to the same shared variable owned by any *home* VCore are seen by all remote caches in the same order. To ensure that all VCores see the *same* global order of Atoms, each *home* VCore keeps a counter called *global_order* maintained by DSMLayer. This counter indicates the number (order) that the next Atom (which may contain load/store instructions to this *home* VCore's shared variables) will be tagged with. This counter is initialized to 1. Each VCore also maintains

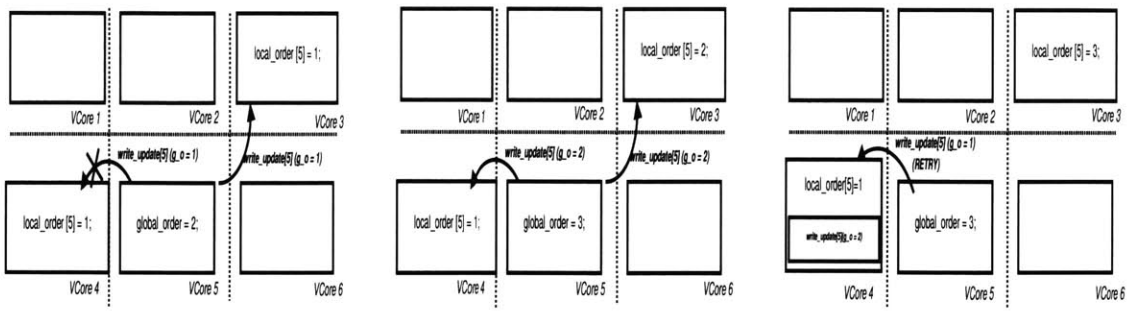


Figure 5-1: Walkthrough example of SRCC for two writes to VCore 5. Only VCores 3, 4 ,5 are detailed for clarity.

a *local_order*, which indicates which number (order) this VCore will accept next, also initialized to 1. A VCore accepts a write update when the *global_order* of the write update equals its current *local_order*, and subsequently increments *local_order*. Write updates with higher orders are buffered until their turn arrives. Figure 5-1 walks through one such transaction of SRCC.

5.3 DIPLOMA Implementation

5.3.1 DIPLOMA's API

Table 5.1 lists the DIPLOMA API. First, the application programmer wishing to use DIPLOMA implements the UserApp *i.e.* the service to be provided in the network. Within the UserApp, the programmer implements the function bodies of the Atoms that can be executed on any specified *home* VCore at run time. Atoms can contain arbitrary Java code that may contain reads and writes on multiple variables on one home VCore. The application logic in the UserApp requests the execution of an Atom by calling a method exposed by DSMLayer, `makeAtomRequest`. Behind the scenes, the DSMLayer routes the request to the specified home VCore, where `handleAtomRequest` is invoked with a reference to the local portion of shared memory on which to execute the Atom. `handleAtomRequest` (implemented by the programmer) returns a reply which is routed back to the originating VCore and passed to `handleAtomReply` (also implemented by the programmer). The programmer may

Table 5.1: DIPLOMA API Methods

Method	Implemented by → Called by	Invoked on	Description
long **makeAtomRequest**(long atomId, long destVCoreX, long destVCoreY, boolean isWrite, byte[] data);	DSMLayer → Programmer	Requesting region	Request to execute a predefined Atom (identified by atomId) on a destination VCore. Can include data. Returns a long to identify the request.
Atom **handleAtomRequest**(DSMLayer.Block b, Atom c);	Programmer → DSMLayer	Target region	Execute an Atom on the local portion of shared memory and return a reply Atom.
void **handleAtomReply**(Atom a);	Programmer → DSMLayer	Requesting region	Callback for receiving an Atom reply.
void **atomFence**(long destVCoreX, long destVCoreY);	DSMLayer → Programmer	Requesting region	Block until all pending Atoms have finished at the destination region.

also call `atomFence` to block program execution until all pending and in-flight Atom requests to a home VCore from a requesting VCore have either succeeded or failed / timed-out.

5.3.2 Prototype Design

We implemented DIPLOMA as an Android application running on Nexus S phones with 3G and Galaxy Note phones with 3G and 4G. Our implementation is comprised of 3 components: the application-developer-implemented app (UserApp), which runs on top of the DIPLOMA Shared Memory Layer (DSMLayer) with caching (SRCC) (enabled optionally), which runs on top of the Virtual Cores layer (VCore). All 3 components run in a single thread to eliminate inter-thread communication. This also ensures execution of Atoms cannot be interrupted by VCore protocol messages. Atoms are also marked with Java's `synchronized` keyword to disallow concurrent access.

A second thread runs a busy-wait loop to receive packets on the adhoc WiFi interface. To communicate between the first and second threads, a Mux is implemented in a third thread, so packets can always be en/dequeued regardless of activity in the first thread. When a VCore needs to upload shared memory to the cloud server, the VCore layer pauses the DSMLayer, serializes the shared memory to JavaScript Object Notation (JSON), and sends it over the cellular network to the server.

5.3.3 Practical Considerations

Next, we discuss some of the issues that arise in a practical deployment of DIPLOMA, describe how our implementation deals with them and continues to operate correctly.

Wireless range more limited than assumed. DIPLOMA's default behavior for VCore assumes that the exiting leader remains in wireless range of its old region when it moves to a neighboring region, so that it can elect a new leader. If the old leader moves out of range before electing a new one, it sends its state to the cloud server so that a new node may download and boot the VCore later. If the wireless range turns out to be much smaller than expected, it could cause many region reboots, hurting latency and completion rate. Our benchmark deployment (Subsection 5.4.1) shows that WiFi wireless range is sufficient: 57% of leader hand-offs succeed without requiring a region reboot, enough to achieve completion rates up to 95.3%.

Resilience to node failures. DIPLOMA monitors for low battery or user opt-out, and initiates leadership hand-off. It also monitors for unexpected node failures with a leader-to-cloud heartbeat (every 120 seconds in our implementation), so that the server will become aware of node failures and allow a new node to become the leader with the last known state.

Atomic execution of Atoms in the face of interrupts. In our implementation, the DSMLayer runs in the same thread as the VCore layer and message handling methods are marked with Java's `synchronized` keyword to ensure that VCore protocol messages cannot interrupt Atom execution. Additionally, when the VCore layer hands off leadership, it pauses the DIPLOMA layer, ensuring that no DIPLOMA Atom requests are processed by the old VCore while or after the new VCore receives the state. Instead, any DIPLOMA Atom requests received during the hand-off a dropped and resent to the new VCore later by the requesting VCore.

Intermittent cellular connectivity. When a node needs to make a cellular access, e.g. upon entering an empty region, it sends a request to the server to become the VCore, retrying if the server is unreachable. Thus, for DIPLOMA to work, the cellular connection must be eventually available. Current metropolitan cellular

networks exhibit this behavior; in our benchmark deployment (Subsection 5.4.1), 3G was available 98% of the time.

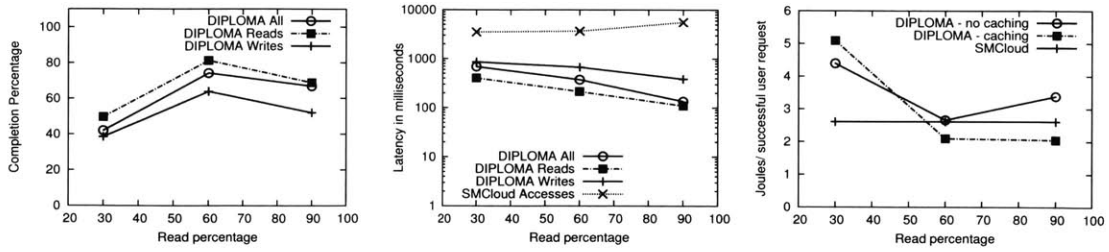
5.4 Evaluating DIPLOMA

We implemented two mobile applications to evaluate DIPLOMA vs cloud-only solutions: a synthetic benchmark that is scripted to generate a specified percentage of read and write requests to a random VCore, and a Panoramio-like [143] app. For comparison, we also implemented cloud-only applications functionally equivalent to the DIPLOMA versions, but relying purely on HTTP requests over 3G/4G to a single-threaded Python web server. The server ensures that accesses to the shared memory are consistent, and provides the same functionality. The server is located in the same geographic region as the phones to minimize backbone Internet latency.

5.4.1 Benchmark App

We carried out a deployment with our synthetic benchmark running on Google Nexus phones with 3G radios in a covered pavilion last year. The area is divided into four regions of 5mx5m per region. Ten volunteers held two phones each, with DIPLOMA running on one phone and cloud-only shared memory (SMCloud) on the other. The volunteers walked among the regions with the phones and indicated which region they were in at a given time. We evaluated DIPLOMA under combinations of SRCC caching disabled/enabled and varying read/write distributions. We measured DIPLOMA's performance against the cloud-only version (SMCloud) using: (1) average latency of successful requests, (2) completion rate of requests, (3) average energy consumed per successful request, and (4) cellular data consumption. Our methodology and results are detailed below.

Average latency: User interface interactions are timestamped to obtain end-to-end request latencies. We compare DIPLOMA to SMCloud in Figures 5-2b and



(a) Completion w/ caching (b) Latency w/ caching (c) Wireless total power

(d) Completion w/o caching (e) Latency w/o caching (f) Wireless active power

Figure 5-2: Completion rate, latency and power comparison of SMCloud and DIPLOMA in Pedestrian Deployment

5-2e[5]. Request latencies for DIPLOMA are typically an *order of magnitude lower* than those in SMCloud.

Without caching, read and write latencies do not vary greatly across read vs. write distributions, as they both incur hops to remote HOME VCores. With caching enabled, high read percentages (90%) show significantly decreased latencies: when requests are serviced at the local VCore from its cache, hops to remote VCores can be eliminated. Write latencies are significantly higher than read latencies because they require write updates to be broadcast to the entire system. This increased write latency is even more pronounced at lower read (higher write) percentages (60%, 30%) as the write updates increase network congestion, and even impact and increase read latencies, too. Thus, caching is advantageous in applications with a higher proportion of requests being reads.

Request completion rate: We calculate the percentage of issued requests that complete (Figures 5-2a and 5-2d). Again, we measure reads and writes separately and in aggregate, and compare the completion rate of DIPLOMA to SMCloud.

[5]SMCloud results appear in both the cache and no cache trials because we ran it in every trial simultaneously against DIPLOMA to control for cellular conditions between trials.

Without caching, the completion rate of application-level requests is 57%, and does not vary between read/write distributions, as expected. With caching, at 60% reads, *80% of application-level requests on DIPLOMA complete.* Note that these application-level requests incur an extra wireless hop from a client app to the UserApp on the region's VCore, which may fail before DIPLOMA is even invoked; the completion rate of the DIPLOMA Atoms alone is 90.9% for 60% reads, and *95.3% for 90% reads.* Caching allows many read requests to be successfully serviced from the local VCore even when a read request to the remote VCore fails.

The completion rate is lower at lower read distributions (30%) due to several factors: more requests are writes, which have lower completion rates than reads because they cannot be cached and must be sent to remote regions; higher wireless contention due to more write updates being broadcast to the entire network, resulting in dropped application packets. This is seen in the disparity between DIPLOMA-level and application-level request completion rates. The application-level implementation does not implement a retry/ack mechanism, unlike DIPLOMA. Thus, at 90% reads, though 95.3% of the DIPLOMA Atoms successfully complete at the VCore, the local VCore's subsequent reply to the client node is only received in 66.8% of requests.

In contrast to DIPLOMA, in SMCloud we observe a 100% completion rate (not shown in figure) of requests, but requests can take as long as 55 seconds to complete in our evaluations. Such high latencies are instances of a problem called Bufferbloat [156]. We discuss DIPLOMA's completion rate further in Section 5.4.3.

Power consumption: We use the Monsoon power meter [108] to build an energy model for the Nexus S devices. Devices running DIPLOMA use adhoc WiFi, so energy for access point scanning and associations is not incurred. Consistent with previous studies [142, 157], our results shows that the energy of a WiFi transmission is significantly less than that of 3G. In our applications, a single HTTP request over 3G is measured to consume 2.6 Joules, while a single WiFi packet transmission might consume only 0.066 J. We do not factor into account energy expended in localisation because this is a task common to both SMCould and DIPLOMA.

We create a linear regression for receive and transmit energy across several packet

sizes (1k, 2k, 4k, and 8k bytes) (R-squared=0.999 for Tx, 0.959 for Rx). This regression is applied to average packet sizes calculated from the deployment logs to obtain per-packet energies for each of the deployment trials, obtaining total energy consumed by WiFi and 3G in each trial.

WiFi idle power (turned on, but not receiving or transmitting) is also measured, and then calculated for each of the trials using experimental run time. Again, consistent with [142, 157], we find that WiFi consumes significant idle power: with only the 3G radio turned on, current consumption is 149 mA. Once adhoc WiFi is turned on, current consumption increases 46% to 218 mA, without any WiFi traffic.

We use these observations to measure the power consumption of both DIPLOMA and SMCloud by processing logs offline. Both SMCloud and DIPLOMA applications wait for 2 seconds between requests[6]. The 3G radio does not return to a low power state between requests in SMCloud due to cloud accesses being much more frequent; therefore, measurements include 3G tail energy [142] for all cloud accesses. Taken together, these measurements give total energy consumed by WiFi + 3G for DIPLOMA, and total energy consumed by 3G for SMCloud, per trial. These totals are then divided by the number of successful requests per trial to arrive at an average energy consumed per successful request per trial.

As we see in Figure 5-2f, DIPLOMA *reduces active wireless energy consumption by up to 94%* per successful request. However, when WiFi idle power is factored in, DIPLOMA is more energy efficient only with caching enabled at 60% read distributions or higher (Figure 5-2c), due to WiFi idle power being quite significant. This highlights the need for better power management of WiFi radios when used in adhoc mode for short-range phone-to-phone communications.

Cellular access reduction: SMCloud solely communicates with the cloud server over the cellular data network, so a cloud access is incurred for every read or write request to shared memory. In contrast, DIPLOMA incurs cloud accesses only for region bootups and leadership changes, which occur due to mobility rather than

[6]2 seconds being a realistic time between user interactions. We choose not to batch requests since they are user-initiated, and to maintain a responsive user experience, should not be delayed

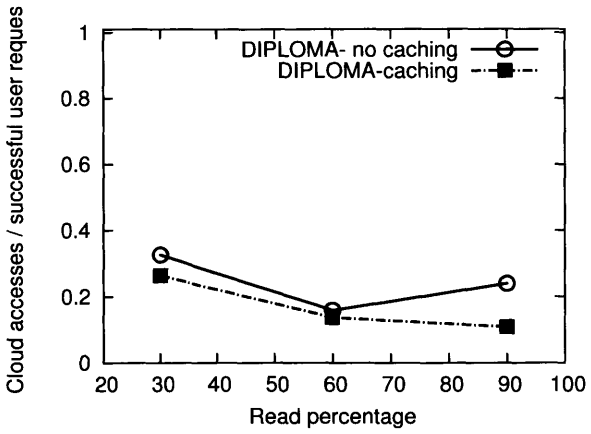


Figure 5-3: Number of cloud accesses

application interactions, so these accesses are amortized over the requests from the application. Hence, we divide the total number of successful cloud accesses by the number of successful requests (DIPLOMA was able to reach the cloud through 3G in 98% of attempts). These results are shown in Figure 5-3 where the x-axis represents the percentage of reads in our benchmark app.

DIPLOMA without caching averages 0.21 cloud accesses per successful request, a 79% reduction from SMCloud, and DIPLOMA with caching averages 0.14 cloud accesses per successful request, a 96% reduction. Caching leads to more successful requests and quicker responses, while the number of cloud accesses remains the same. This advantage is more pronounced at higher read percentages.

5.4.2 Panoramio-like App

We implemented a Panoramio-like app on Galaxy Note phones to demonstrate that popular consumer mobile apps today can be readily ported onto DIPLOMA. In the app, we use the shared memory abstraction provided by DIPLOMA to retrieve and update photo data. Users (clients) can *take* pictures of interesting things where they are, and they can also *get* pictures taken by other users. The photos are stored in the same region that they are taken in. If a user desires to view photos from a remote region, *gets* can traverse multiple hops on their way to a remote region. The phones serve double duty by both participating in DIPLOMA (as leaders or non-leaders) and

being the clients of the application themselves. To reduce the size of data transfers, we apply JPEG compression to all pictures before transmission. We also implement a functionally equivalent cloud version (CCloud) of the same app (accessed through 3G/4G) and compare the DIPLOMA version without caching (CameraSM) to the cloud based version in terms of completion rate and request latencies.

We carried out a deployment of Panoramio on 20 Galaxy Note phones over 3G and 4G networks this year, with 10 phones running CameraSM, and another 10 running CCloud. Phones are placed statically and uniformly across 6 regions (5mx5m each) within an open indoor space. Two people walk around the phones clicking on buttons simultaneously on CameraSM and CCloud pairs of phones, taking and getting pictures. We present mean and median latencies in Tables 5.2 and 5.3, omitting distributions for brevity. Similar to the benchmark application, we also measured the number of cloud accesses per application-level *get* or *take* request for both CameraSM and CCloud. Since CCloud makes a cloud access on every request, this number is 1 for CCloud on both 3G and 4G networks. For CameraSM, we observed 0.29 cloud accesses per request on 3G, and 0.22 accesses per request on 4G . Since the phones were static, these accesses were primarily due to leader-to-cloud heartbeats which occurred at 2 minute intervals. The heartbeat interval allows us to trade off between number of cloud accesses and the reboot time of an unpopulated region. We observed a high completion rate of 98.6% for CameraSM across 573 requests, and 100% for CCloud across 564 requests. These results show DIPLOMA outperforming both 4G and 3G cloud implementations in response times while retaining high completion rates.

As Panoramio has substantial write traffic, our write update caching protocol leads to excessive WiFi traffic (approximately 6KB write updates for every region when a picture is taken, plus associated ACKs) and was turned off in this deployment. In hindsight, applications like Panoramio would work better with a write-back protocol. We don't have power comparisons for Panoramio as 4G has more sophisticated power management, making it difficult to apply a power model naively to our activity traces to get accurate power estimates.

Table 5.2: Panoramio-like app latencies over 3G

	takes CameraSM	*takes* CCloud	*gets* CameraSM	*gets* CCloud
mean	144 ms	2558 ms	217 ms	2279 ms
median	109 ms	2465 ms	161 ms	2229 ms

Table 5.3: Panoramio-like app latencies over 4G

	takes CameraSM	*takes* CCloud	*gets* CameraSM	*gets* CCloud
mean	144 ms	546 ms	178 ms	469 ms
median	107 ms	534 ms	159 ms	469 ms

We also conducted outdoor mobile deployments with this app, but saw high loss rates over ad-hoc Wifi, which could be due to the large packet size of images, high WiFi interference in the area, and/or poor antennas on the Notes. We are in the process of diving further into these ad-hoc WiFi problems and investigating potential optimizations.

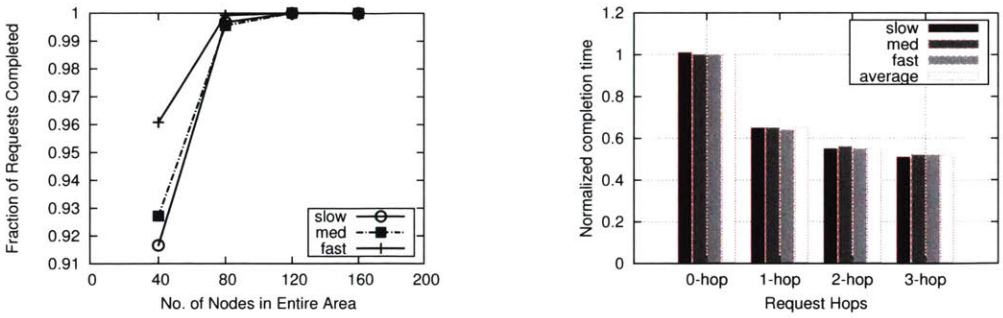
5.4.3 Simulation studies

We use ns-2.37 [148], a discrete event network simulator, to evaluate our system at scale with the synthetic benchmark. Node mobility is simulated with the Random Way Point model with three settings: slow, medium and fast (Figure 5.4). Node movements are constrained to a $350m \times 350m$ terrain and the radio range is fixed at 250m. 250m is well within the transmission range of 802.11p or DSRC [158], which we expect will become the basis for adhoc communications for distributed mobile apps. This radio range dictates our region size since every broadcast has to be heard by the neighboring regions as well, resulting in 4 x 4 regions. Since we have 4 regions in each dimension, we also evaluate the efficiency of caching for requests that traverse between 0 and 3 hops. Each simulation lasts 40000 seconds.

Variation of node density. We vary the number of nodes from 40 to 160 to study the effect of increasing node density on DIPLOMA's performance. The resulting node density is close to typical car densities in US cities which vary from 1700-8000 cars per square mile [159], or about 80-380 cars for our $350m \times 350m$ terrain. Figure 5-4a shows the effect of varying the number of nodes on the completion

Table 5.4: Simulation settings

Parameter	slow	med	fast
Min. speed (m/s)	0.73	1.46	2.92
Max. speed (m/s)	2.92	5.84	11.68
Min. pause time (s)	400	200	100
Max. pause time (s)	4000	2000	1000
Mean cross time (s)	48	24	12



(a) Completion rate w/o caching (b) Request completion latency

Figure 5-4: Completion rate and latency of DIPLOMA in simulation.

rate of DIPLOMA. We see that increasing the node density significantly improves the performance of DIPLOMA. Also, after a threshold density of 80 nodes, the completion rate saturates near 100%.

Usefulness of caching. One intuitively expects caching to be more useful for reads to farther away regions. Writes would also take longer since they trigger updates in SRCC. To study this, in Figure 5-4b we plot the completion time of a request with caching enabled for varying node speeds. The numbers are normalized to a no-caching implementation. The proportion of reads and writes is kept equal to avoid any bias. We see that caching improves latency for all requests spanning 1 hop or more. On average, the 1-hop, 2-hop and 3-hop requests have a 35%, 45% and 48% lower request latency as a result of caching. However, the incremental benefit of caching decreases with increasing hops. This is understandable since write latencies scale linearly with hop count.

In summary, our simulation results demonstrate the effectiveness of caching and show how penetration of DIPLOMA affects performance. We envision that a large city scale deployment will have sufficient density to achieve a completion rate close to 1, while simultaneously providing the latency and cellular utilization benefits we

observed in our deployments.

5.5 Related Work

DIPLOMA is related to several systems in Computer Architecture, Sensor Networks, Distributed Algorithms and Distributed Systems. We outline key similarities and differences.

Computer Architecture: Most commercial architectures, such as x86 [160] and IBM PC [161], stay close to sequential consistency [162] by reordering only certain instruction combinations. Similar to DIPLOMA, some processor architectures (Alpha [163], Sparc [164]) aggressively reorder all instructions by default and provide memory fences for the programmer or compiler to enforce ordering if required.

Among research systems, DIPLOMA is closest to Release Consistency (RC) [149]. RC defines memory operations as either *ordinary* or *special*. *Special* operations are either synchronization or non-synchronization accesses. Synchronizing accesses are either *acquire*s or *release*s. Memory accesses within an *acquire-release* block form a critical section and execute atomically, provided each critical section is protected with enough *acquires*. Every **Atom** in DIPLOMA implicitly begins with an *acquire* and ends with a *release*, guaranteeing exclusive access to the Atom's shared variables.

DIPLOMA has similarities to Transactional Memory [165]: Atoms are like transactions, but transactions allow atomic modifications to arbitrary portions of the memory, while Atoms operate on memory belonging to one VCore alone.

Sensor Networks. Several programming languages have been proposed for collections of resource-constrained devices. Kairos [166], an extension of Python, abstracts a sensor network as a collection of nodes which can be tasked simultaneously within a single program. Pleiades [167] borrows concepts from Kairos and adds consistency support to the language. These proposals are tailored to static sensor nets and do not deal adequately with mobility.

Distributed Algorithms. Most distributed algorithms for mobile agents tackle programmability by first emulating a static overlay. Virtual Nodes (VN) [145] is

one such abstraction. Section 5.2.1 discussed the practical issues with VN. Geoquorums [168] provides consistency support using a quorum-based algorithm to construct consistent atomic memory over VNs, but it assumes reliable physical layer communication. [169] presents complex algorithms to implement reliable VNs over an unreliable physical network through consensus, which is expensive in practice on wireless networks.

Distributed Systems. There are several loosely coupled distributed systems that explore varying notions of consistency. Bayou [170] allows eventual consistency between data copies residing on differing replicas, which could be mobile nodes or dedicated servers. All replicas are equal and merged opportunistically using an anti-entropy protocol. In contrast, DIPLOMA maintains one authoritative copy of the data (the VCore) and actively resolves conflicts using cache coherence. CODA [171], is a file system for mobile devices with unreliable cellular connections. DIPLOMA instead targets shared memory and assumes modern cellular connections are far more reliable (albeit with very long and variable latencies). InterWeave [172] is a hierarchical consistency model with varying consistency guarantees for different levels ranging from hardware shared memory to weakly consistent shared memory across the Internet. It is significantly different from our system since DIPLOMA is homogeneous and flat and operates primarily on wireless LAN links. Semantically, TreådMarks [173] is the closest to DIPLOMA since it implements release consistency. Further, similar to DIPLOMA, it implements Distributed Shared Memory. However, TreadMarks is tailored to a workstation environment with highly reliable LAN links. Mobility and network unreliability are new problems DIPLOMA tackles.

Chapter 6

Device-centric sensing: Enabling pervasive autonomy for ITS

This chapter contains work presented at the International Conference on Robotics and Automation (ICRA) May of 2016, titled "A smartphone-based laser distance sensor for outdoor environments."[1]

This chapter presents a new mobile sensor with a device-centric approach: a smartphone-based laser distance sensor that is novel in its combination of low-cost and suitability for outdoor environments, to enable promising future ITS of pervasive mobile robots and self-driving vehicles. The device-centric design leverages existing mobile imaging and processing, and some additional low-cost commodity hardware, to robustly identify laser illumination in the presence of ambient light, a critical challenge for low-cost distance sensing systems. We evaluate the performance of our full hardware-software system prototype with outdoor experiments with vehicular scenarios. We also demonstrate a full-system prototype of an autonomous outdoor mobile robot, enabled by our sensor.

[1]

6.1 Introduction

Mobile robots must avoid obstacles when navigating an environment. This is typically done with a laser distance sensor (LDS), which is also often used for localization and mapping [28, 29]. Work on passive camera-based systems can eliminate the need for expensive LDS devices, such as LIDAR, for localization [174], but an LDS is still useful for reliable obstacle detection and avoidance. The high cost and complexity of LDS devices, however, has precluded their use for low-cost applications and handheld use.

We present Smartphone LDS, a low-cost multi-point laser distance sensor, based on a smartphone, that works outdoors. It is designed to leverage off-the-shelf components and the rapid improvement and proliferation of phones with low-cost, high performance image capture and processing. Our prototype, shown in Figure 6-1, combines a phone with an off-the-shelf line laser module, and leverages the phone's camera, processor, and input/output to simultaneously measure multiple distances across a planar field-of-view. By utilizing the processing power of the phone, we can perform more intensive image processing to identify the laser illumination and reject ambient light, improving performance for outdoor use. Our sensor has the characteristics shown in Table 6.1. To our knowledge, there is no other outdoor 2D laser distance sensor that combines a low-cost laser illumination source with computer vision-based image processing techniques.

Smartphones are increasingly pervasive, and are continuously and rapidly increasing in computing power and sensing capability. This has not gone unnoticed in the robotics community: even Robot Operating System (ROS) is available on Android [175], developed by Google and Willow Garage.

Compared to the typical laser distance sensors utilized in autonomous outdoor robots, smartphones are also low cost: the Android phone used in our system, a Nexus 5, has an original retail price of $349, and the hardware additions for our prototype cost less than $50, as shown in Table 6.2. We believe the additional hardware costs can be significantly reduced in mass production. Thus, adding high-resolution distance

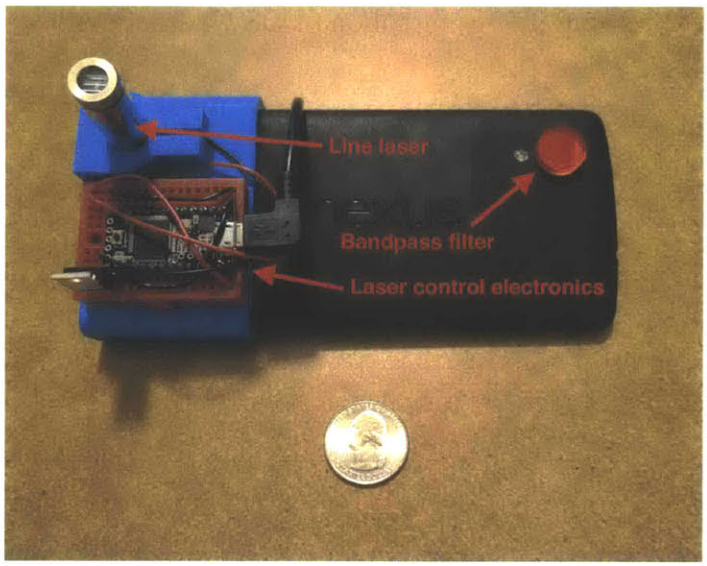


Figure 6-1: Smartphone LDS with major components labeled. United States quarter for size comparison.

sensing to the phone enables a complete robotics platform, including the laser distance sensing and ROS software environment that many researchers are accustomed to, at low cost and in a mobile form factor.

A phone-based LDS enables many applications that were not previously possible, due to its pervasive deployment and low cost. In Section 6.4, we evaluate our system in the scenario of obstacle detection and avoidance in autonomous vehicles. Other potential applications include obstacle avoidance on lightweight personal mobility vehicles, 3D scanning with phones, navigational aids for the visually impaired through smartphone apps and small autonomous robots utilizing advanced localization and mapping algorithms designed for laser distance sensors.

There are several distinguishing characteristics of our design from prior work on low-cost laser distance sensors such as the Revo LDS [176] by Konolige *et al.* In particular, our design:

- **Works outdoors**, as shown in our evaluation in direct sunlight in Section 6.4.1. We tested the Revo LDS design outdoors by extracting the unit from a Neato XV-11 robotic vacuum cleaner, and were unable to obtain range measurements beyond half a meter when the target surface was illuminated by direct sunlight.

Table 6.1: Smartphone LDS Characteristics

Works in sunny outdoor conditions under strong ambient light
Solid-state design with no moving parts
Field-of-view of 48 degrees (dependent on camera and laser lens)
Returns range readings up to 2m in direct sunlight, 5.8m indoors
Fast: 14400 readings / second (480 simultaneous readings at 30 Hz)
High minimum angular resolution of 0.1057 degrees
6 cm range error at 5 m
Low cost: Leverages pervasive smartphones & inexpensive off-the-shelf components
Eye-safe

Table 6.2: Smartphone LDS Cost Breakdown

Nexus 5 smartphone	$349
Optical bandpass filter	$13
Line laser	$22
Control electronics	$12
3D-printed mount	$2
Total cost of hardware additions	**$49**
Total cost including smartphone	**$398**

The Revo LDS utilizes bandpass filtering and temporal synchronization of the laser illumination with a global shutter CMOS image sensor to reject ambient light, but sunlight is still too strong for it to reliably discriminate the laser light. Our design modulates the laser illumination and leverages relatively powerful heterogenous processing cores on smartphones to perform image-processing across multiple camera frames, improving ambient light rejection.

- **Has a solid-state design** with no moving parts. This makes it lower-cost and more suitable for handheld and/or consumer applications, with simple assembly, no gyroscopic effect from spinning parts, and no moving parts to break or wear out. This is unlike other laser distance sensors that typically use a spinning mirror or electronics to scan the laser and take measurements.

- **Exploits silicon advances readily.** Smartphones are pervasive, and have been rapidly increasing in computing power and camera performance. This means the performance of the system design will improve, and only relatively simple hardware modifications / attachments to the phone are required. For example, better cameras and processors in newer phones will improve the throughput, range resolution, angular resolution, and detection latency and/or lower cost.

6.2 Background and related works

Addressing the general problem of distance sensing is important for mobile robots that must navigate unstructured environments. We discuss several typical approaches below:

- Laser distance sensors such as LIDAR have the advantage of high spatial resolution, allowing robots to discriminate between multiple types and sizes of obstacles, but its cost is prohibitive for lower-cost systems due to the need for high-speed circuitry for accurate time-of-flight ranging, high-powered laser diodes and photodetectors, and electromechanical components to scan the optics over the field of view.

- Stereo-vision has the advantage of not requiring active illumination, but relies on complicated and intensive computation, and performs poorly in environments that have surfaces lacking textures for the stereo correspondence algorithm to exploit.

- Ultrasound is low cost and commonly used on small mobile robots, but suffers from low range and low spatial resolution [177] that is inadequate for object discrimination and classification.

- Radar has the advantage of measuring an object's relative speed to the sensor in real-time by leveraging the doppler shift effect, but also suffers from poor spatial resolution across the field of view and high cost.

Our design's key contribution is that it is low-cost and works outdoors in sunlight. We use an active triangulation approach. Below, we discuss several related approaches in other active illumination distance sensors.

- **Pulsed Time-of-Flight (ToF).** Systems operating under this principle constitute what is typically considered LIDAR, and include devices such as the SICK LMS 291 and Velodyne 3D LIDAR sensors [30] that are often used in autonomous vehicles research. ToF sensors emit a very short, high-power laser pulse, and measure the time it takes to detect its reflection from a distant surface. The high-power lasers, sensitive photodetectors, high-speed electronics, and scanning optics found in these systems result in prohibitively high cost. Kimoto et. al. [178] developed a 3D LIDAR that is relatively low-cost compared to 3D LIDARs by adding a resonant mirror to a 2D LIDAR, but it still requires moving components and is high cost compared to our approach.

- **Modulated light Time-of-Flight.** SoftKinetic [179] and Kinect for Xbox One [180] both emit high-frequency (10s of MHz) modulated light and measure the phase shift of the return signal due to the time of flight to provide a 3D depth image. However, their ambient light rejection is not strong enough for outdoors use due to the wide divergence of illumination energy both horizontally and vertically. These approaches also require specialized CMOS imagers and control circuitry in order to capture and process signals at high speed.

- **Structured light.** Several commercially available depth sensors utilize structured light, measuring distortions in a projected pattern to determine depth, including Google Project Tango [9] and the original Kinect [181]. Similarly to the modulated light sensors above, they do not perform well outdoors due to ambient light quickly overpowering the sensor's illumination as the energy diffuses in both directions.

- **Active stereovision.** Intel's RealSense R200 [182] is a compact stereovision depth camera that utilizes patterned illumination to project noise and improve

performance on surfaces lacking textures indoors. It requires a relatively powerful computer, however, and customers report that it does not perform well outdoors [183].

- **Active triangulation.** Our design falls under this category of sensors, which use an illumination source and an arrayed image sensor to locate reflected illumination in the image and calculate the distance. Such systems have similar challenges detecting the active illumination in ambient light, which we address in our design.

Our system is novel in its use of line laser modulation, image processing, and the full use of a 2D image sensor to enable compact, low-cost 2D outdoor laser distance sensing with no moving parts. Our line laser beam diverges in only one axis, maintaining illumination flux at longer distances compared to depth cameras with illumination that diverges in two axes. Other laser distances sensors achieve greater ranges with highly-collimated beams that do not diverge in either axis, at the cost of requiring mechanical scanning, which increases cost and is contrary to our goal. To improve our range performance, we temporally modulate our illumination to increase ambient light rejection.

The Revo LDS [176] is similarly low-cost and also uses active triangulation, but mechanically scans its single-point sensor over the field of view and does not perform well outdoors (Section 6.4.1). Our design samples multiple points simultaneously without moving parts. Quigley et. al. [184] also use a line laser for active triangulation, but the system is not suitable for outdoor use.

6.3 Rangefinder Design

Our system is an outdoor, active-triangulation laser distance sensor. It consists of an illuminator and a detector separated by a baseline distance. The illuminator is a laser module with a line lens, and projects a horizontal beam of laser light into the scene. The detector is the camera on the off-the-shelf Android smartphone, capturing

images to be processed by the phone's processors (CPU and GPU).

The vertical baseline distance between the camera and the line laser emitter causes laser illumination reflected off objects in the scene to be detected at different vertical positions across the image plane of the camera, depending on the distance to the illuminated object. Thus, each column of pixels in the image corresponds to one measurement within the field-of-view, and we can simultaneously take as many measurements as there are columns of pixels (480 in our prototype, as discussed in Section 6.3.2).

These major components of our system are shown in Figure 6-1, while a detailed diagram of the hardware and software components of our system is shown in Figure 6-4.

6.3.1 Challenges

There are several competing demands on this system:

- **Eye safety.** The laser needs to be eye-safe due to its outdoor operation, which limits its output power and the maximum range of detection.
- **Processing performance.** The sensor needs to perform image processing from the camera to distinguish the signal of interest (the line laser return) from the background (other objects in the scene, background radiation from the sun, etc.). This processing needs to be done in real-time on a low-cost device.
- **Sensing fidelity.** We have limitations on camera frame rate, dynamic range, and electronic rolling shutter (due to the CMOS image sensor in the phone).

We discuss the impact of possible improvements to our system in Section 6.6.

6.3.2 Design Criteria and Characteristics

Because our design leverages several off-the-shelf components, our design must consider the properties of the system that we cannot change:

- **Phone processor performance.** Figure 6-2 shows the per-frame total processing time of our image processing kernels versus image resolution. Running at the full frame rate of the camera (30 fps) gives us 33.33 ms to process each frame, so we capture images at 640x480 pixels resolution, down-sampled from the full resolution of the camera, to stay within the limits of the phone's compute performance.

- **Camera frame rate.** The maximum frame rate of the camera is 30 frames per second.

- **Camera sensor physical characteristics.** The camera on the Nexus 5 phone has a focal length of 3.97 mm and physical sensor dimensions of 4.6032 by 3.5168 mm, as reported by the Android Camera2 API. As we are using the longer dimension of the sensor to localize the laser illumination along, each pixel in the down-sampled image corresponds to a physical distance on the sensor of 0.0072 mm.

- **Available baseline separation.** The size of the phone provides a lower bound on the size of the overall system, but an overly large baseline will result in a large system.

The triangulation geometry described in Konolige *et al.* for the Revo LDS also applies to our system, with perpendicular distance to an object from the baseline separation as

$$q = \frac{fs}{x} \tag{6.1}$$

with x the position of the reflected laser illumination on the CMOS imager. To enable comparison of our Smartphone LDS to Revo LDS, we wish to achieve an fs product comparable to the $fs = 800$ of [176], and thus must maximize our baseline separation, as our focal length is fixed and smaller than that of the Revo LDS. We choose our baseline separation to be 155mm, slightly longer than the length of the phone, in order to keep the system compact, with the laser attached rigidly to the phone via

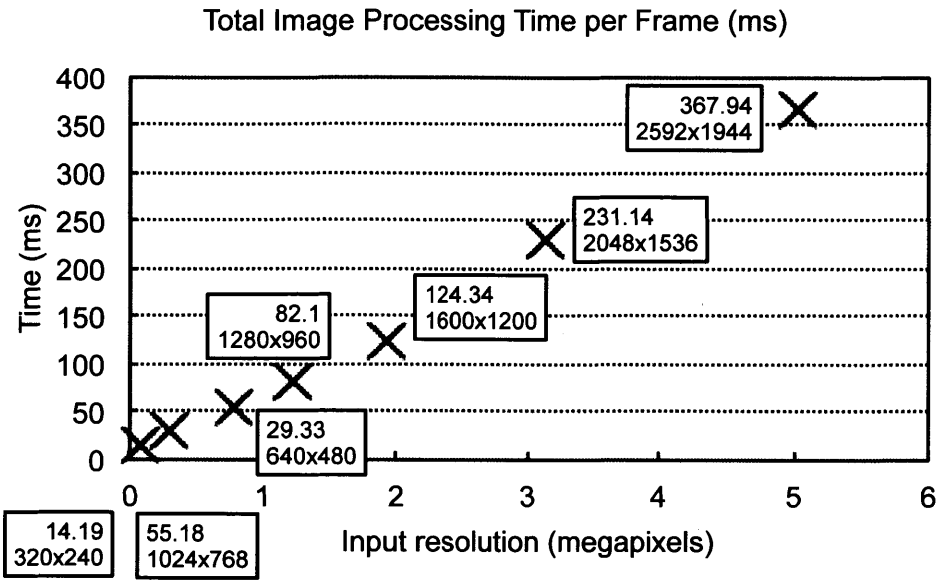


Figure 6-2: Total processing time per frame by our app versus image resolution.

a 3D-printed mount. The resultant $fs \approx 615$ provides us with comparable range resolution, minimum distance, and size as the Revo LDS.

To provide readings across the field of view, we use the narrower vertical dimension of the image sensor to triangulate the laser light at multiple angles simultaneously. Equation 6.2 gives the position y of the laser illumination on the image sensor, and

$$\tan\theta = \frac{y}{f} \tag{6.2}$$

Thus, our angular sensitivity is dependent on the position y on the image sensor and the physical camera characteristics:

$$\frac{d\theta}{dy} = \frac{f}{f^2 + y^2} \tag{6.3}$$

where dy is the width of a pixel: 3.5168 mm / 480 pixels. Angular resolution is lowest in the center of the field of view, with an angular resolution of 0.1057 degrees, and highest at the edge, with an angular resolution of 0.09513 degrees.

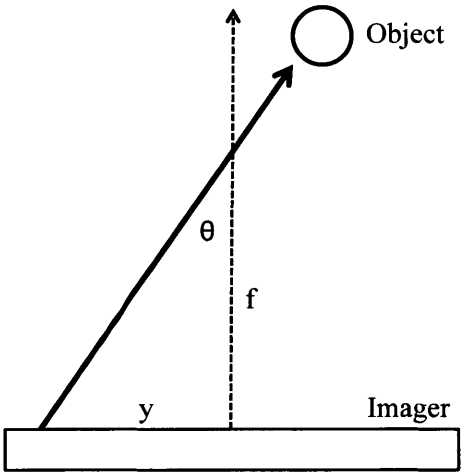


Figure 6-3: Geometry of angle to object. The angle θ is calculated from the y position of the laser illumination in the image.

6.3.3 Ambient Light Rejection

Typically, the use of a line laser will result in the laser energy being spread out over the line, making it difficult to detect the reflected illumination at longer distances. We add a 20nm bandpass filter to reduce most of the ambient light flux. We also modulate the laser and perform additional processing on the phone to further reject ambient light.

The modulation of the laser is controlled by the smartphone through a microcontroller connected via the Android USB Host API. The laser is pulsed on alternate frames. Due to the electronic rolling shutter on the camera, the laser pulse must last for the entire frame capture duration of 1/30 s, rather than just the image exposure time.

The app tracks per-pixel luminosity transitions between frames and counts how many occur on-off or off-on as expected from knowledge of the laser pulse modulation. When the number of matched transitions detected reaches a threshold (4 in our experiments), the pixel is tagged as having the modulated laser illumination present. Requiring a number of matched transitions imposes a latency penalty of $threshold/framerate$ seconds before a new range reading can be detected. A threshold of 4 imposes a latency of 0.133 seconds, comparable to that of the Revo LDS, which cannot detect an object's distance until it has physically scanned across it. At

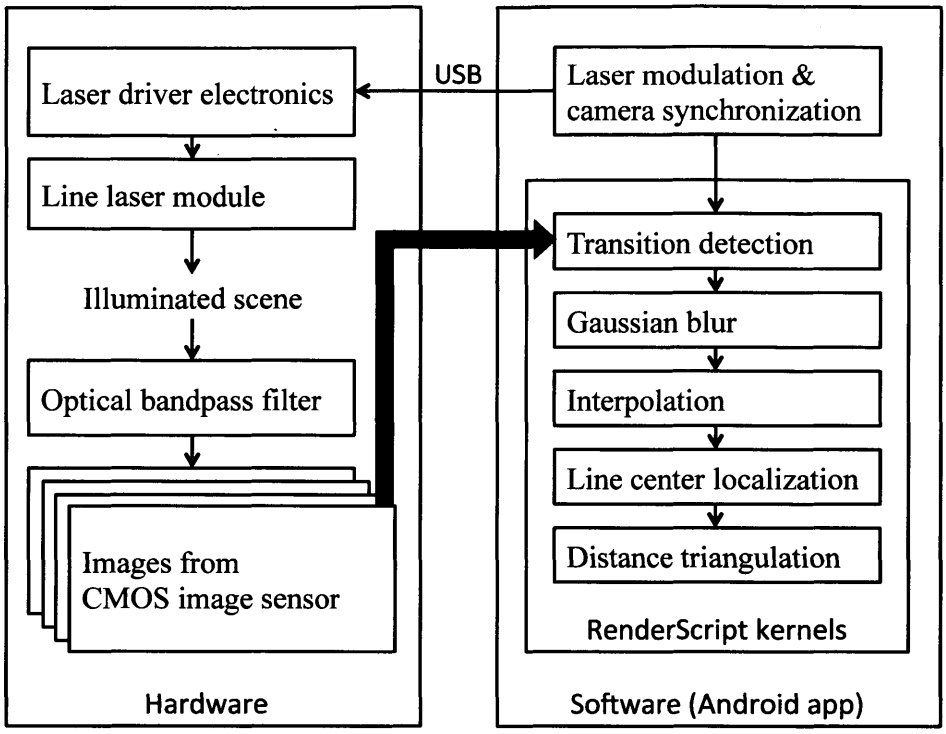


Figure 6-4: Block diagram of hardware and software components.

a 5 Hz scan rate, the latency of the Revo LDS can range from 0 to 0.2 seconds.

This algorithm provides good rejection of ambient noise; Figure 6-5 shows the input image on the right, and the detected laser illumination on the left, highlighted by the blue boxes. The phone is able to process images at 30 Hz, the maximum framerate of the camera. The image processing is accelerated on the CPU and GPU through the use of RenderScript, Android's parallel computing framework.

6.3.4 Eye-Safety

To prevent distraction to passers-by during our outdoors and vehicle experiments, we chose to use a non-visible wavelength of 780 nm for the illumination. and removed the infrared cut-off filter from the smartphone camera. We add a 20 nm optical bandpass filter to further reduce ambient light, as discussed above. The 100 mW line laser is effectively pulsed at 15 Hz with a 50% duty cycle. The lowest Maximum Permissible Exposure (MPE) is $5.8 \times 10^{-5} J/cm^2$ given by the equation for a repetitively pulsed laser [185]:

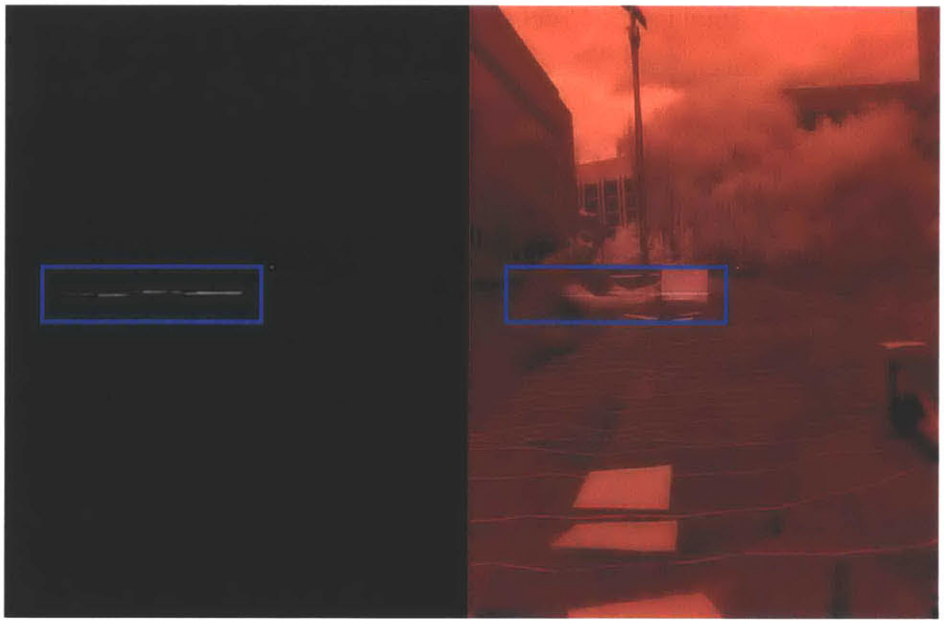


Figure 6-5: Right: input image. Left: algorithm output.

$$\frac{1.8C_a t^{0.75} \times 10^{-3}}{n^{1/4}} \tag{6.4}$$

where $C_a = 10^{2(0.78-0.7)}$, $t = 1/30s$, and $n = 150$ pulses in 10 s (natural motion of the eye). The power absorbed by the retina if looking directly into the laser beam falls off quickly with distance due to the very wide 60 degree horizontal divergence. For safety, we assume all of the energy within the vertical divergence of the beam falls on the retina of the eye and we do not include energy loss through the air. Thus, the minimum eye safe distance or Nominal Ocular Hazard Distance (NOHD) at which the MPE is not exceeded is 1.45 m, given by:

$$A \times MPE = Pt \frac{w}{2D \tan(divergence/2)} \tag{6.5}$$

where $A = 0.385cm^2$ is the area of the pupil, P is the pulse power, D is the NOHD, and $w = 0.7cm$ is the pupil width (with all of the beam's vertical width falling on the pupil).

There are multiple ways to accomplish this: the system can include an interlock mechanism that reduces the power or disables the laser when a closer object is

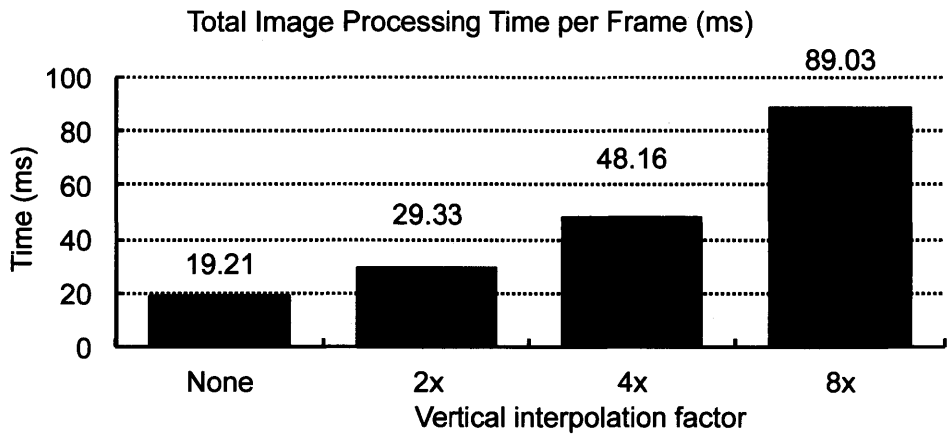


Figure 6-6: Total processing time per frame vs. interpolation.

detected, or for integration into larger systems such as vehicles, a physical shroud around the system can prevent exposure closer than the NOHD. Alternatively, with global shutter sensors, which are becoming available in commodity smartphones (Section 6.6), the pulse duration could be reduced from 33.33 ms to the exposure time of 3 ms, increasing the MPE by 5.6x, reducing NOHD to 0.26 m.

6.3.5 Laser Line Localization

To localize the laser line within the image, we first gaussian blur the image to spread out saturated pixels and reduce noise to better locate the center of the line in each column of pixels, and then interpolate the image of the line 2x to improve subpixel accuracy along each column. The interpolation is limited by the processing time of our unoptimized prototype software, and affects our range resolution at longer distances. Figure 6-6 shows total image processing time for other interpolation factors at an input resolution of 640x480. After interpolation, we find the center of the laser light in the row using the maximum value. Finally, to improve the subpixel localization accuracy, we calculate the centroid with the 8 neighboring pixels on each side of the center.

6.3.6 Calibration

The range errors of the system primarily come from:

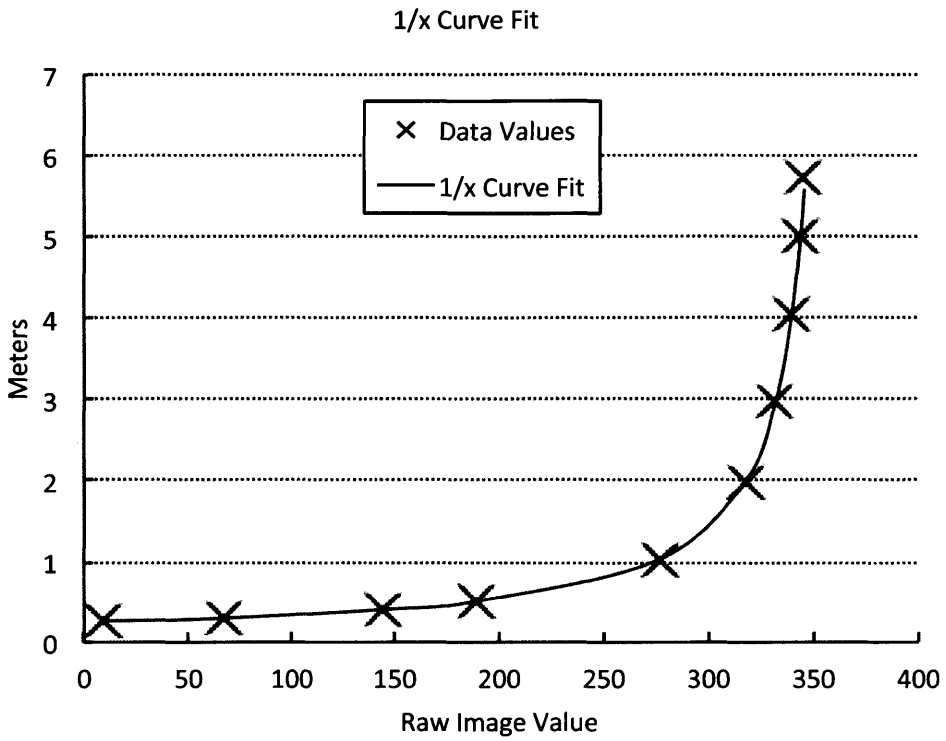


Figure 6-7: Calibration readings and 1/x curve fit.

- **The angle between the laser and the camera.** This comes from inaccuracy in the 3d-printed mount, inaccuracy of the low-cost line laser diode alignment, and misalignment and distortion from the line lens on the laser. Due to the use of a line laser with a slightly greater horizontal divergence than the field-of-view of our camera, the system can tolerate misalignment (up to +/- 6 degrees) of the laser within the plane of the emitted line, as a rotation of the laser within that plane still results in pixels across the field-of-view being illuminated with reflected laser light. We rely on manual alignment of laser module around its longitudinal axis and perform a single calibration that is applied to all rays, but a per-ray error calibration would better reduce systematic error across the field of view.

- **Residual lens distortion.** The known lens distortion is corrected for by Android's camera software subsystem, but there may be residual distortion due to manufacturing and alignment variation in the camera module.

Similarly to [176], we address these errors by calibrating a 1/x curve fit (Figure 6-

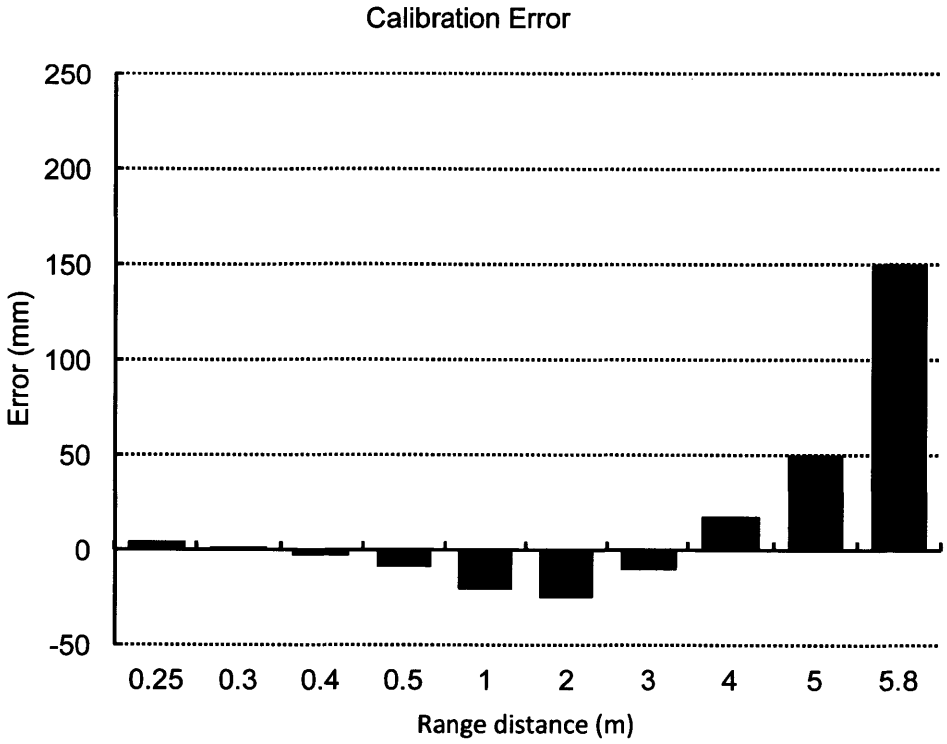


Figure 6-8: Calibration error after 1/x curve fit.

7) between our localized laser positions in the raw image and the distance to a white (greater than 90% reflectivity) calibration target, with more weight given to longer distances because the higher slope results in greatly magnified localization errors. We use an LDS from a Neato XV-11, which is based on the device from [176], to provide our baseline of known distances. The curve fit is imperfect, and errors remain after calibration, shown in Figure 6-8. To correct for remaining error, we use table of offsets from the actual distance, applied with interpolation above distances of 1.0 m, which eliminates the effects of calibration errors, but subsequent mechanical flex of the 3d-printed mount could cause further calibration errors to manifest.

6.4 Performance Evaluation

We evaluated our phone-based laser distance sensor by first characterizing its distance sensing performance outdoors, and then characterizing the system in an example application scenario: obstacle detection for an outdoor autonomous vehicle.

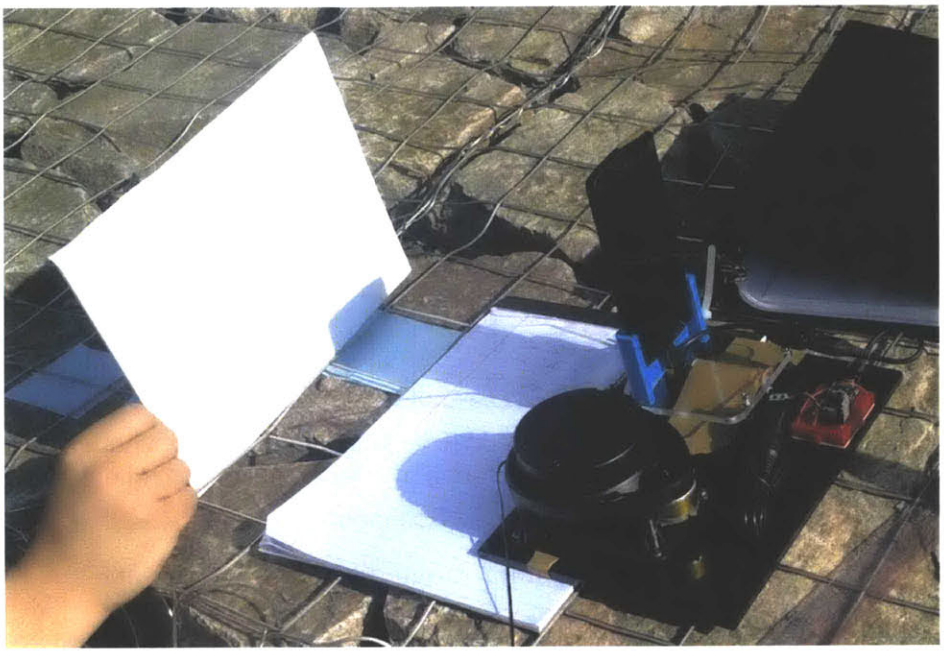


Figure 6-9: Experimental setup of Smartphone LDS vs. XV-11.

6.4.1 Outdoor Distance Sensing Evaluation

We tested Smartphone LDS outdoors by mounting it next to the XV-11 device, depicted in Figure 6-9, and measured range error vs. distance for white targets (90% reflectance) and black targets (10% reflectance) under the following scenarios:

- Outdoors, with target surface shaded from sunlight.
- Outdoors, with target surface under direct sunlight.

Figure 6-10 shows the maximum range achieved by each system outdoors in our measurements. Smartphone LDS is consistently able to detect further targets than the XV-11.

Figure 6-11a shows the total error for white and grey targets. We also include the measurements from the calibration for comparison, performed indoors with a white target. Figure 6-11b shows the XV-11 unit tested under the same conditions for comparison, but we do not have the calibration data for it. Measured errors are comparable to those in [176].

Our standard deviation error remains under 10 mm indoors for all ranges, and

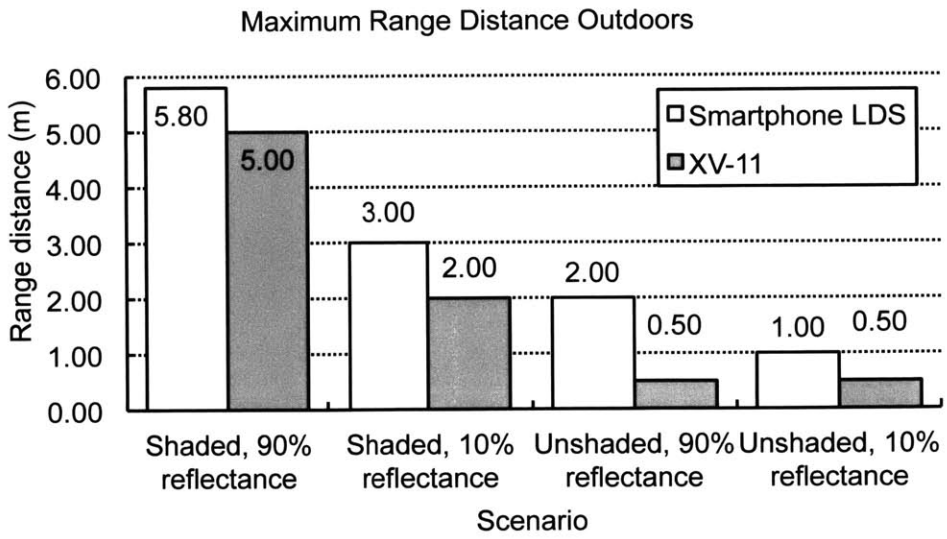


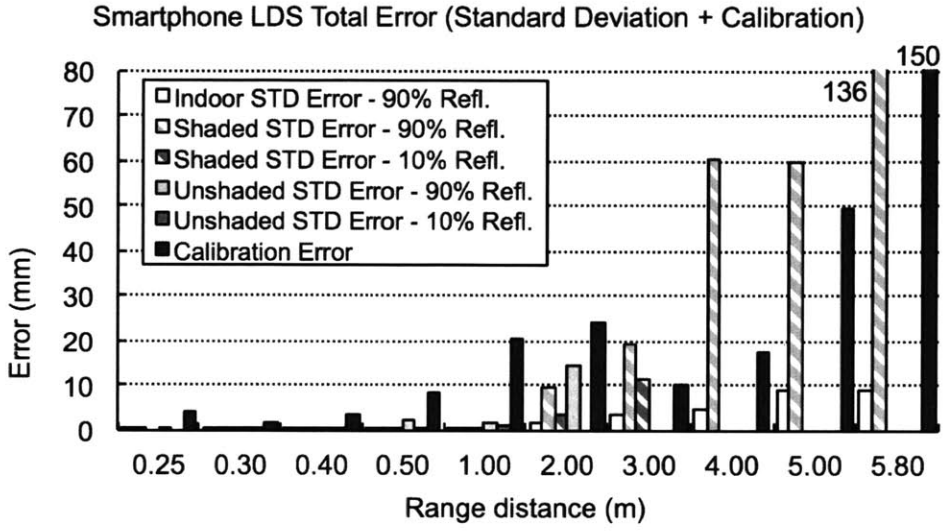
Figure 6-10: Maximum range distance comparison.

under 2 mm outdoors to 1 meter. At 3 meters, Smartphone LDS was unable to detect targets under direct sunlight.

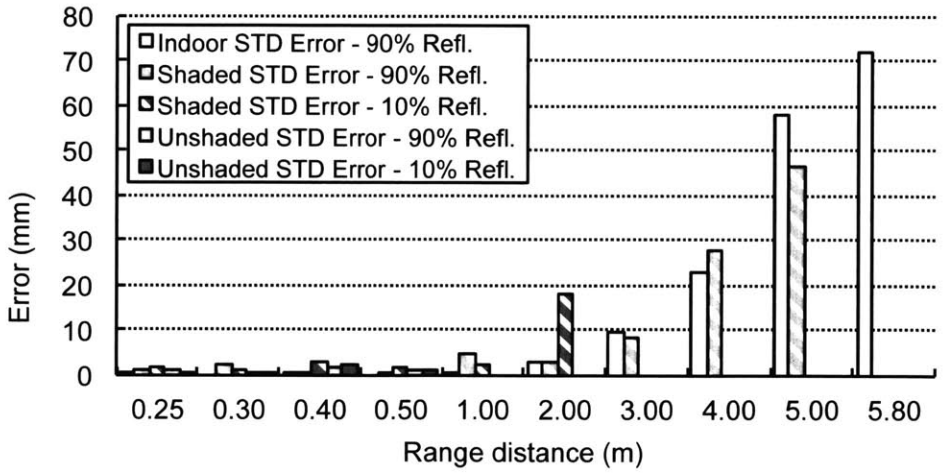
Smartphone LDS was also unable to obtain range readings at 0.25 m for black targets: the low reflectance combined with undesirable vignetting at the edge of the image caused by our undersized off-the-shelf bandpass filter such that the system could not find bright enough reflected laser light. In our outdoor scenarios, there was unintentional movement of the targets due to wind and hand movements contributing to standard deviation error for both devices.

6.4.2 Obstacle Detection Evaluation

We also evaluate our system in an example scenario of obstacle detection for a low-speed autonomous vehicle, mounting it on the front bumper of the testbed vehicle in [29] that is also equipped with a SICK LMS 291 LIDAR sensor, used as a baseline. The LMS 291 provides a 180 degree field of view, performing measurements at 75 Hz with 10 mm range resolution, +/- 15 mm range accuracy, and 0.25 degree angular resolution. The range of the LMS 291 and other similar laser distance sensors is many tens of meters, an order of magnitude more than our sensor's range, putting them in a different class in both cost and performance. Even with a much more limited range,



(a) Smartphone LDS error under test scenarios.



(b) XV-11 LDS error under test scenarios.

Figure 6-11: Total error comparison.

Figure 6-12: Vehicular mount for system and experimental setup.

our sensor is still useful for obstacle detection for low-speed autonomous vehicles, and/or for other low-cost mobile robots where the cost of a sensor like the LMS 291 is prohibitive.

Thus, we evaluated the system in three common obstacle avoidance scenarios, illustrated in Figure 6-13.

1. Obstacle (pedestrian) moving towards and away from front of vehicle.
2. Obstacle (pedestrian) crossing in front of vehicle.
3. Vehicle moving towards and away from stationary object (cardboard box).

Table 6.3 shows the minimum distance (out of 5 trials) at which each object was detected in each of the scenarios. We assume instantaneous application of the brakes upon detection and a comfortable deceleration of $a = 3.4m/s^2$ for collision avoidance [186]. The detection distance is our available vehicle braking distance, and we determine the maximum speed the vehicle can operate at and still avoid collision in Table 6.3 with Equation 6.6 relating stopping distance and constant deceleration.

$$v = \sqrt{2ad} \tag{6.6}$$

In our experiments, there was significant rolling shutter distortion and vibration in the image due to vibration of the vehicle, to which Smartphone LDS was rigidly

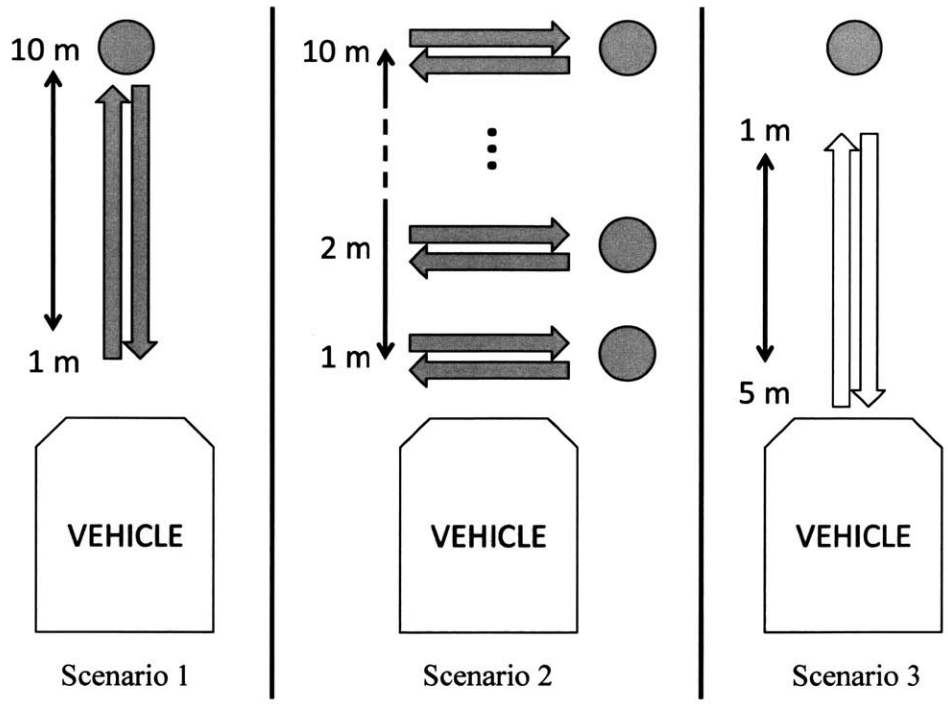


Figure 6-13: Diagram of obstacle avoidance scenarios.

attached, which impaired transition detection and significantly reduced the range compared to Section 6.4.1. In comparison, the LMS 291 was able to detect the obstacle at all distances tested. Despite the adverse effect of the rolling shutter distortion, Smartphone LDS is still able to provide sufficient performance for obstacle avoidance at speeds ranging from 14.8 to 18.5 km/h in the scenarios.

Table 6.3: Stopping Distance and Corresponding Speed

Scenario	Detection distance	Collision can be avoided under
1	2.5 m	14.8 km/h (4.1 m/s)
2	2.8 m	15.7 km/h (4.4 m/s)
3	3.9 m	18.5 km/h (5.1 m/s)

6.5 Full System Demonstration

In addition to our performance evaluations of Smartphone LDS, we also demonstrated our sensor providing the primary sensing for a simple autonomous robot by integrating

Figure 6-14: Smartphone LDS mounted on a self-balancing scooter.

it with Smartphone LDS. Significantly, this demonstration robot shows that the sensor makes it possible to build an outdoor-capable autonomous vehicle for under 1000 USD.

6.5.1 Integration with vehicular platform

We chose a *Xiaomi Ninebot mini* self-balancing scooter (similar to a Segway self-balancing scooter) to serve as the vehicle, due to its designed use in outdoor environments, and its ability to be remote-controlled by a manufacturer-provided mobile Android app via Bluetooth.

We mounted Smartphone LDS on top of the Ninebot scooter, as depicted in Figure 6-14. Since the Ninebot scooter can be controlled via a mobile app over Bluetooth, and Smartphone LDS is based on a smartphone capable of running Android apps, we used a Bluetooth packet sniffer to reverse-engineer the Bluetooth commands sent

from the manufacturer's mobile app to the Ninebot, and modified our Smartphone LDS Android app to use these commands to connect to and control the Ninebot wirelessly via Bluetooth.

6.5.2 Demonstration behaviors

In our Android app, we implemented two autonomous behaviors that depend on the distance sensing of Smartphone LDS:

1. An object following behavior that maintains a pre-determined distance away from an object in front of the robot.

2. An obstacle avoidance behavior continuously drives the robot forward, and if an obstacle is detected in front of the robot within a specified range, stops and turns the robot around to avoid colliding with it.

We demonstrated these two behaviors and the robot to the general public at the Massachusetts Institute of Technology 100 Open House event, which was held outdoors on the MIT campus. Figure 6-15 shows several frames of a video of the robot executing the object following behavior: the robot moves backwards and forwards in order to maintain a pre-determined distance from a person walking towards and away from the robot. Figure 6-16 shows several frames of a video of the robot executing the obstacle avoidance behavior: the robot successfully approaches an obstacle, stops in front of it, and then turns away from it.

In both example behavior scenarios, the smartphone did all of the computation for both the distance sensing and for the autonomous behaviors. We also implemented a web-based remote control interface for control of the robot via the Internet. The ability to implement several key subsystems of a mobile autonomous robot via a single device showcases the advantages of using a smartphone to serve as the brains of an autonomous robot. As smartphones and mobile devices improve in processing, wireless communication, and sensing capabilities, a smartphone-based autonomous robot can rapidly leverage those advances.

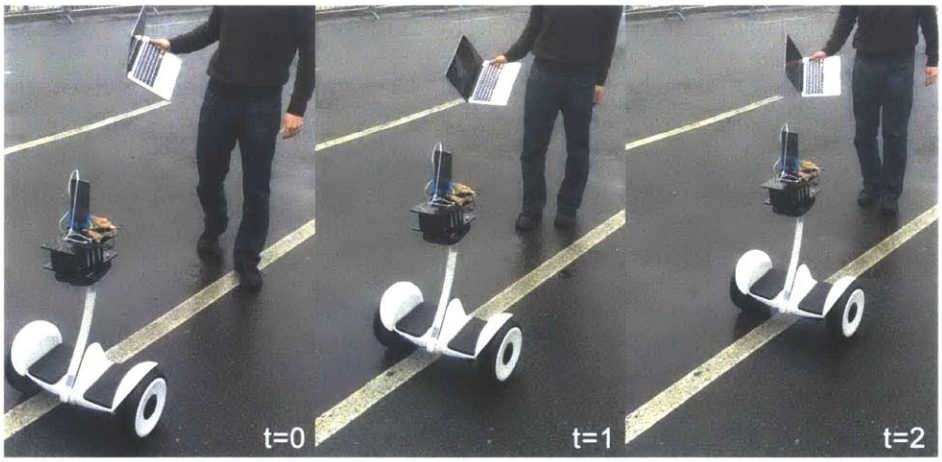


Figure 6-15: Video frames showing robot following a person.

6.6 Discussion and Conclusion

Our range resolution, angular resolution, maximum range, and detection latency are limited by the performance of the camera sensor and processor performance. The rapid improvement of mobile cameras and processors is the key motivation behind the design of Smartphone LDS, and we discuss the potential improvements that could be realized.

6.6.1 Smartphone camera

The CMOS image sensor on the Nexus 5 constrained some aspects of our system performance. To improve it, we could use sensors with the following features beyond the limits of the Android device with:

- **Global shutter.** A global shutter would allow reducing our laser pulse duration from the rolling shutter duration, 33.3 ms, to the exposure time, typically 0.5 to 3 ms. This improves range by allowing shorter and stronger laser pulses and/or improved eye-safety hazard distance. Global shutter sensors targeting computer vision are beginning to appear in mobile devices [187], such as the Amazon Fire Phone [188].

- **High frame-rate.** This reduces detection latency and improves rejection of other periodic signals that might naturally occur in the scene. Many newer

Figure 6-16: Video frames showing robot avoiding an obstacle.

phone cameras already support high frame-rates up to 240 frames per second, a 12x improvement, and our exposure time is already low enough to allow these high frame-rates.

- **High dynamic-range.** This prevents sensor saturation by ambient light such that the laser illumination is not detectable, while still allowing us to detect weak laser reflections at long range or on low-reflectance surfaces.

6.6.2 Smartphone processor

Our input image resolution (640 x 480) is well below the full sensor resolution (3264 x 2448) due to processing bottlenecks, limiting range resolution and angular resolution. Compared to the Snapdragon 800 processor in the Nexus 5, the recent Snapdragon 810 processor has 113% more CPU performance [189] for a parallelized Sobel kernel (representative of our kernels), and 280% more GPGPU performance [5]. Since our image processing is already written as highly parallelized kernels, we estimate that we could at least double the image processing resolution in each dimension, doubling minimum angular resolution to 0.05287 degrees as in Equation 6.3 and doubling range resolution dq by halving dx in the range sensitivity equation [176]:

$$\frac{dq}{dx} = \frac{q^2}{fs} \tag{6.7}$$

6.6.3 Security

Attacks that replay recorded laser illumination have been demonstrated on commercial LIDAR systems [190]. Smartphone LDS controls the laser modulation on a per-frame basis, so it could use an unpredictable/pseudorandom modulation sequence (with sequence length affecting latency as in Section 6.3.3), providing a defense against maliciously replayed laser illumination.

6.6.4 3D distance ranging and multi-user ranging

Using an alternate modulation sequence can also enable discrimination of multiple lasers in the field-of-view, enabling multiple planes of depth to be sampled, while rejecting unknown signals from other nearby LDS systems. Using multiple laser illumination planes restricts the resolution of the additional dimension, but maintains high illumination flux. The Velodyne 3D LIDAR similarly uses multiple laser illumination planes.

6.6.5 Conclusion

We presented a smartphone-based laser distance sensor that is low-cost, solid-state, compact, and works outdoors. The only modifications required to the phone were the addition of a line laser and driver electronics connected to the phone via USB, and a replacement of the camera's infrared-block filter with a band-pass filter. The appearance of infrared-sensitive and global shutter cameras on mobile devices, along with advances in phone processors, will enable Smartphone LDS to achieve improved range, resolution, and latency, and continue to readily work with unmodified phones by simply attaching a protective phone case with integrated electronics, laser, and filter.

Chapter 7

Conclusion

In this chapter, we review the main contributions in Section 7.1 and discuss future research directions in Section 7.3.

7.1 Thesis Summary

We began with the observation that mobile devices are increasingly pervasive and capable, with many new transportation services enabled by the widespread adoption of mobile devices. We then explored some specific challenges in realizing next-generation intelligent transportation systems, and how new technological capabilities could address them. Next, we reviewed the current trends in key processing, sensing, and communications technologies in vehicular and smartphone contexts. To realize rapid adoption and uptake of these new technologies, we emphasized a *device-centric* approach that leverages the rapid turnover and upgrade-cycle of mobile devices instead of the slow upgrade and deployment cycle of transportation vehicles and urban transportation infrastructure.

In Chapter 3, we presented a device-centric wireless communications radio, based on an existing vehicular standard, that enables device-to-device (D2D) communications for mobile systems.

In Chapter 4, with RoadRunner, we identified a motivating challenge for future ITS: pervasive traffic congestion management. We leveraged D2D communications

and mobile sensing to design, prototype, and evaluate a traffic management system that reduced traffic congestion compared to an existing state-of-the-art system, and eliminated the need for physical traffic management infrastructure.

In Chapter 5, with DIPLOMA, we harnessed the power of D2D communications and rapidly improving mobile processing performance to run ITS services on collaborating networks of mobile phones, demonstrating improvements in latency, power, and cellular bandwidth usage.

In Chapter 6, with Smartphone LDS, we presented a novel laser distance sensor to enable future ITS incorporating pervasive self-driving cars and other autonomous mobile robots, again leveraging the increasing performance and sensing capability of mobile devices. Our system prototype was able to realize a critical sensing modality for autonomous robots at low cost, while performing in challenging outdoor conditions.

Collectively, the research presented in this thesis represents several key processing, sensing, and communications technologies to enable future ITS scenarios that require highly responsive urban networks and applications, autonomous urban vehicles and robotics, and urban sensing working in concert. These future scenarios can be realized sooner rather than later with a device-centric approach, revolving around the ubiquitous mobile phone.

7.2 Perspective and lessons learned

In the course of this thesis, we also discovered new challenges, and shortcomings of the approaches that we took. Below, we distill and summarize some key takeaways, and how we might do things differently if we were to revisit the projects in this thesis with newfound experience:

- **The impact of importance of reducing protocol overhead for device-to-device system architectures.** With DIPLOMA, we implemented a distributed protocol for data storage and cache coherence for general computation. The results indicated that caching could significantly improve performance in

some situations, but negatively impacted performance in others. This was due to the high overhead of the cache update protocol in the context of mobile ad-hoc networks. Even with the improved latency of device-to-device communications compared to infrastructure-centric communications, one must still be careful when allocating the latency budget to various portions of communications protocols. After our experience with DIPLOMA, we designed a much simpler protocol for RoadRunner, but it was not as generalizable. A redesign of the coherence protocols in DIPLOMA might consider modifying the protocol design to strike a balance between number of communications round-trips required and generalizability.

- **The superlinear value of wireless communications range for device-to-device apps.** To first-order, the number of nodes reachable by any one node in a geographic region scales quadratically with the communications range, as area scales with the square of radius, since in the work presented, nodes were constrained to the surface of the earth (in an aerial context, this effect becomes even more pronounced as volume scales with the cube of radius). We were limited to small deployment areas in DIPLOMA's experimental evaluation, which was our earliest project, due to the use of standard Wi-Fi radios with limited communications range. With RoadRunner's experimental evaluation, we expanded our deployment area significantly, and also measured superlinear impact that an improved communications range had in enabling and proving out the benefits of using a device-centric, V2V approach: the range improvement Wi-Fi to 802.11p was 6x, but V2V offload improvement was 9.4x. In conclusion, for piecemeal adoption of device-centric technologies, an increase in communications range can net you even greater gains, both in system performance and in making it easier to reach critical mass.

- **The need for open hardware interfaces in mobile computing.** As above in RoadRunner, the addition of longer-range wireless communications hardware to a mobile phone significantly improved the performance of the system and

enabled the protocols we designed to be used in more realistic experiments. This required effort in interfacing Android devices to hardware not designed for that purpose, with relatively inelegant workarounds. In general, a lot of our effort was dedicated to working around the limitations of mobile device hardware. For example: kernel module hacks to interface with D2D radios, abusing USB tethering to interface to an FPGA, USB driver modifications to meet timing constraints for the laser distance sensor, and modifications to the Android operating system to support ad-hoc Wi-Fi. Furthermore, some of the hardware we introduced is only useful to a user in certain contexts, and thus may not warrant being integrated into a mobile device permanently, and may be better deployed as a module that can be attached when necessary. If we were to work on these projects again, we would first build a high-bandwidth, low-latency interface for hardware to be attached to the mobile device, so that we can reuse that effort across multiple projects. Indeed, this has just recently come closer to reality, as mentioned in Section 7.3.5.

- **The value of real-time experiment-support software.** A small but useful component in all of our projects was the testing and evaluation code; specifically, bits of code and user interfaces that we often reused or re-implemented across projects to set up repeatable experiments, manage entire fleets of device in real-time during experiments, and view sensor data in real-time to ensure experiments were working properly. This code was not directly useful for our research contributions, but supported the real-world evaluation of our work. For future work that involves real-world deployments, we would find it very valuable to again invest effort in this kind of supporting software.

7.3 Future work

As mobile devices continue to increase in on-device processing performance and device-centric communications capabilities, and cellular networks continue to become increasingly congested and expensive, we foresee more and more ITS services being

hosted on collections of collaborating mobile devices, rather than relying on centralized infrastructure to tackle challenges beyond those presented in this thesis. Furthermore, as transportation vehicles also become increasingly dependent on mobile-driven technologies to deliver new features and capabilities, mobile devices such as smartphones can provide a viable path for upgrading existing vehicles with new features and capabilities and future-proofing against the need to upgrade vehicles themselves.

Smartphones are the pervasive mobile device today, but the advantages of a device-centric approach to ITS can extend to new mobile device form factors and capabilities. Furthermore, as more and more ITS systems adopt a device-centric approach and reduce their reliance on infrastructure, new challenges will emerge in ensuring security, privacy, and trust in these widely-dispersed systems, enforcing regulations, scaling device-centric communications, and reducing power consumption so that yet more pervasive and constrained devices can support new ITS services. In this section, we discuss some of the opportunities for future research on mobile device-driven intelligent transportation systems.

7.3.1 Security, privacy, and trust

Security, privacy, and trust must be maintained in future ITS and cyber-physical systems to encourage adoption and deployment, and a balance must be achieved between allowing future ITS to access personal information to improve and customize services for a user, while protecting that personal information from unnecessary or malicious access. Furthermore, much of these specific needs and trade-offs will depend on the specific ITS applications and services, and existing approaches may be reconsidered in the context of transportation, such as privacy-preserving self-hosted proxies for mobile devices [191], or answering mobile location and interest-based user queries without allowing attackers to reconstruct private user information through decentralization [192].

As a user's personal smartphone becomes part of collaborating, networked ITS apps, approaches such as mobile device sandboxing and virtualization [193] might provide coarse-grained isolation between publicly-accessible ITS services and personal

computing needs, and future work might include systems to provide more granular access from sandboxed computation for ITS to personal information on user devices.

7.3.2 Policy and regulatory considerations

ITS services often have numerous parameters that can be adjusted to affect system behavior, and future work must deeply consider challenges in how to bridge the gap between technological capability and desired system behavior for transportation. RoadRunner, as a motivating example, already prompts future research directions of how transportation policy interacts with new technologies. For example, with any traffic management system, transportation designers and city planners must decide what metrics they want to prioritize (e.g. travel time versus throughput). For RoadRunner, this means how RoadRunner should request tokens, how many tokens to allocate to roads, and which roads to put under congestion control. Beyond adaptation of ITS services to policy and regulatory demands, future work includes how to simulate the effects of new policy and regulation (SimMobility [133] is one example of an urban simulator designed to study strategic policy effects), how to automatically adapt and tune parameters like those in RoadRunner, and how to detect and characterize infractions of regulations by users.

7.3.3 Additional ITS services

This thesis prototyped and demonstrated RoadRunner, a traffic management system, and Panoramio, an urban photo sharing platform, as motivating examples of future ITS and urban services. There are many other challenges in future ITS and new services that could be tackled via similar device-centric approaches that tightly couple hardware and software: detecting and mapping parking availability with new sensors and real-time information sharing between devices on vehicles and pedestrians, information sharing between mobility on-demand services, context-aware apps, and public transportation services to provide just-in-time transitions between different modes of transportation, and are just a few examples.

The time-sensitive and location-aware nature of many transportation services makes them well-suited for location-based real-time information sharing models like DIPLOMA's, and the real-world success of leveraging existing location and inertial sensors for traffic monitoring and activity detection points to the high potential of exploiting new, transportation-targeted sensors for future ITS services.

7.3.4 New sensors

As described earlier, smartphones today already contain many sensors, and Smartphone LDS is an example of a new sensor not available in current phones. Future phones could integrate additional environmental sensors beyond barometer (gas sensors for urban pollution monitoring [194], radiation monitoring [195]), additional health sensors beyond heartbeat (ECG for mobile heart monitoring [196], EEG for urban psychology [197]), and even more. As phones incorporate an ever growing and heterogenous collection of sensing modalities, the potential to collect data across large networks of phones and derive actionable information for new ITS and urban services or research becomes very promising.

7.3.5 Modular hardware

New mobile devices are beginning to include modular designs that allow extra hardware to be integrated with mobile phones: Google's Project Ara [198] aims to make a smartphone composed of multiple modules for the processor, screen, storage, camera, etc. For example, the LG G5 smartphone has swappable components for extended functionalities like improved camera controls or higher-quality audio [199], and the Motorola Moto Z exposes a high-speed expansion bus for new modules such as a pico projector, louder speakers, or extended battery [200]. The modularity of next-generation smartphones can enable even more rapid adoption of future ITS technologies: instead of waiting for phone manufacturers and users to adopt specific new hardware like laser distance scanners or D2D radios, modules can be easily and rapidly deployed onto a user's phone. Even the mobile device itself might be treated as a

"module" that can tightly integrated with a vehicle.

7.3.6 New pervasive devices

New computing devices beyond smartphones promise to follow the exponential growth path blazed by personal computers and mobile phones: both the Internet of Things [201] and wearable devices [202] are growing rapidly in adoption and ever-increasing in processing performance and sensing capability [203, 204]. Just like how we exploited the smartphone revolution to build ITS that can be rapidly adopted, we can imagine mapping the ideas demonstrated in this thesis onto these new classes of pervasive computing devices to build device-centric, rapidly-adoptable services for ITS, smart cities, and cyber-physical systems.

7.4 Final remarks

In conclusion, this thesis has presented contributions to mobile connectivity, processing, and sensing capabilities in the context of enabling next-generation intelligent transportation systems that can be rapidly adopted, and prototyped real-world hardware-software systems that leverage these capabilities to tackle urban transportation challenges: managing traffic via improved, device-centric mobile connectivity, providing highly-responsive mobile services via easy-to-use collaborative computation, and enabling pervasive autonomous vehicles via new mobile sensors. By emphasizing the deployment of new technologies into devices, and not infrastructure, these future capabilities can be more rapidly adopted and deployed in our transportation systems, fulfilling the promise of a faster path to increased efficiency, new applications, and wide-spread benefit to our cities and urban transportation systems.

Bibliography

[1] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez. Millimeter wave mobile communications for 5G cellular: It will work! *IEEE Access*, 1:335–349, 2013.

[2] R. Cheng. Verizon to be first to field-test crazy-fast 5G wireless. `http://www.cnet.com/news/verizon-to-hold-worlds-first-crazy-fast-5g-wireless-field-tests-next-year/`, September 2015. (Accessed on 08/24/2016).

[3] B. Klug. Samsung announces big.LITTLE MP support in Exynos 5420. `http://www.anandtech.com/show/7313/samsung-announces-biglittle-mp-support-in-exynos-5420`, September 2013. (Accessed on 08/15/2016).

[4] R. Smith and J. Ho. The Apple iPhone 6s and iPhone 6s Plus review: A9's CPU: Twister. `http://www.anandtech.com/show/9686/the-apple-iphone-6s-and-iphone-6s-plus-review/4`, November 2015. (Accessed on 08/15/2016).

[5] A. L. Shimpi. Qualcomm's Snapdragon 808/810: 20nm high-end 64-bit SoCs with LTE category 6/7 support in 2015. `http://www.anandtech.com/show/7925/qualcomms-snapdragon-808810-20nm-highend-64bit-socs-with-lte-category-67-support-in-2015`, April 2014. (Accessed on 08/15/2016).

[6] T. Litman. Smarter congestion relief in asian cities. *Transport and Communications Bulletin for Asia and the Pacific*, 2013.

[7] A. Davies. This startup says it can make any car autonomous for $10,000. `http://www.wired.com/2014/06/cruise-self-driving-car-startup/`, June 2014. (Accessed on 08/24/2016).

[8] Perrone Robotics. Drop-in autonomy kit. `http://www.perronerobotics.com/dak/`. (Accessed on 08/24/2016).

[9] A. Kadambi, A. Bhandari, and R. Raskar. 3D depth cameras in vision: Benefits and limitations of the hardware. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 3–26. Springer, 2014.

[10] Occipital. The Structure sensor. `http://structure.io/`. (Accessed on 08/24/2016).

[11] S. Woo, H. J. Jo, and D. H. Lee. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, 2015.

[12] J. Motavalli. The dozens of computers that make modern cars go (and stop). *The New York Times*, page B6, 2010.

[13] G. Jerke and A. B. Kahng. Mission profile aware IC design: a case study. In *Proc. Design, Automation & Test in Europe Conference and Exhibition*, page 64. European Design and Automation Association, 2014.

[14] M. Casale-Rossi. Automotive ICs drive advanced design at established nodes. `http://www.techdesignforums.com/practice/technique/advanced-automotive-ic-design/`, February 2015. (Accessed on 08/15/2016).

[15] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 4(3):143–153, 2003.

[16] V. Cerone, M. Milanese, and D. Regruto. Combined automatic lane-keeping and driver's steering through a 2-DOF control strategy. *IEEE Transactions on Control Systems Technology*, 17(1):135–142, 2009.

[17] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[18] S. Curtis. The car of the future is 'the most powerful computer you will ever own'. `http://www.telegraph.co.uk/technology/news/11609406/The-car-of-the-future-is-the-most-powerful-computer-you-will-ever-own.html`, May 2015. (Accessed on 08/15/2016).

[19] Mobileye. Moving closer to automated driving, Mobileye unveils eyeq4 System-on-Chip with its first design. `http://www.prnewswire.com/news-releases/moving-closer-to-automated-driving-mobileye-unveils-eyeq4-system-on-chip-with-its-first-design-win-for-2018-300045242.html`, March 2015. (Accessed on 07/28/2016).

[20] Nvidia. Self-driving vehicles development platform — NVIDIA DRIVE PX. `http://www.nvidia.com/object/drive-px.html`. (Accessed on 08/24/2016).

[21] Intel. BMW Group, Intel and Mobileye team up to bring fully autonomous driving to streets by 2021. `http://newsroom.intel.com/news-releases/intel-bmw-group-mobileye-autonomous-driving/`, July 2016. (Accessed on 07/28/2016).

[22] C. Hughes, M. Glavin, E. Jones, and P. Denny. Wide-angle camera technology for automotive applications: a review. *IET Intelligent Transport Systems*, 3(1):19–31, 2009.

[23] D. Etherington. Tesla's Autopilot 2.0 said to add triple camera system and more radar. http://techcrunch.com/2016/08/11/teslas-autopilot-2-0-said-to-add-triple-camera-system-and-more-radar/, August 2016. (Accessed on 08/15/2016).

[24] A. H. Eichelberger and A. T. McCartt. Toyota drivers' experiences with dynamic radar cruise control, the pre-collision system, and lane-keeping assist. *Transportation Research Board Monograph*, 2014.

[25] K.-T. Song, C.-H. Chen, and C.-H. C. Huang. Design and experimental study of an ultrasonic sensor system for lateral collision avoidance at low speeds. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 647–652. IEEE, 2004.

[26] E. Guizzo. How Google's self-driving car works. *IEEE Spectrum (Online)*, October 2011. (Accessed on 08/15/2016).

[27] M. Harris. Uber could be first to test completely driverless cars in public. *IEEE Spectrum (Online)*, September 2015. (Accessed on 08/15/2016).

[28] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[29] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, E. Rankin, and M. H. Ang Jr. Autonomy for mobility on demand. In *Intelligent Autonomous Systems*, pages 671–682. Springer, 2013.

[30] A. S. Huang, M. Antone, E. Olson, L. Fletcher, D. Moore, S. Teller, and J. Leonard. A high-rate, heterogeneous data set from the DARPA Urban Challenge. *The International Journal of Robotics Research*, 29(13):1595–1601, 2010.

[31] CAR 2 CAR. CAR 2 CAR Communication Consortium. http://www.car-to-car.org. (Accessed on 08/24/2016).

[32] D. Hübner and G. Riegelhuth. A new system architecture for cooperative traffic centres – the simTD field trial. In *Proc. ITS World Congress*, 2012.

[33] Traffic Technology Today. Toyota to introduce V2X communications on 2015 Japanese models. http://www.traffictechnologytoday.com/news.php?NewsID=64353, November 2014. (Accessed on 08/15/2016).

[34] National Highway Traffic Safety Administration. Transportation Sec. Foxx announces steps to accelerate road safety innovation. http://www.nhtsa.gov/About+NHTSA/Press+Releases/2015/nhtsa-will-accelerate-v2v-efforts, May 2015. (Accessed on 08/24/2016).

[35] Cisco. Cisco, NXP invest in Cohda Wireless to enable the connected car. http://newsroom.cisco.com/press-release-content?articleId=1119826, 2013. (Accessed on 08/24/2016).

[36] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, 47(11):84–95, 2009.

[37] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro. LTE for vehicular networking: a survey. *IEEE Communications Magazine*, 51(5):148–157, 2013.

[38] A. Lee. Verizon killing unlimited data plans soon. `http://www.huffingtonpost.com/2011/06/21/verizon-unlimited-data-plan_n_881091.html`. (Accessed on 08/24/2016).

[39] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: vehicular content delivery using WiFi. In *Proc. ACM MobiCom*, 2008.

[40] A. Benslimane, T. Taleb, and R. Sivaraj. Dynamic clustering-based adaptive mobile gateway management in integrated VANET–3g heterogeneous wireless networks. *IEEE Journal on Selected Areas in Communications*, 29(3), March 2011.

[41] Y. Ge, L. Lamont, and L. Villasenor. Hierarchical OLSR – a scalable proactive routing protocol for heterogeneous ad hoc networks. In *Proc. IEEE WiMob.* IEEE, 2005.

[42] L. Zhou, X. Naixue, L. Shu, A. Vasilakos, and S.-S. Yeo. Context-aware middleware for multimedia services in heterogeneous networks. *IEEE Intelligent Systems*, PP(99):1, 2009.

[43] H. Fathi, S. Chakraborty, and R. Prasad. Optimization of mobile IPv6-based handovers to support VoIP services in wireless heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 56(1):260–270, Jan 2007.

[44] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero. Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC? In *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12. IEEE, 2013.

[45] J. Koomey, S. Berard, M. Sanchez, and H. Wong. Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing*, 33(3):46–54, 2011.

[46] M. B. López, H. Nykänen, J. Hannuksela, O. Silvén, and M. Vehviläinen. Accelerating image recognition on mobile devices using GPGPU. In *Proc. SPIE Parallel Processing for Imaging Applications*, pages 78720R–78720R. International Society for Optics and Photonics, 2011.

[47] B. Rister, G. Wang, M. Wu, and J. R. Cavallaro. A fast and efficient SIFT detector using the mobile GPU. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2674–2678. IEEE, 2013.

[48] L. Armasu. Qualcomm introduces next-generation 'Hexagon 680' DSP for Snapdragon 820. `http://www.tomshardware.com/news/qualcomm-introduces-hexagon-680-dsp,29909.html`, August 2015. (Accessed on 08/15/2016).

[49] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[50] P. Klasnja and W. Pratt. Healthcare in the pocket: mapping the space of mobile-phone health interventions. *Journal of Biomedical Informatics*, 45(1):184–198, 2012.

[51] E. Kaplan and C. Hegarty. *Understanding GPS: principles and applications.* Artech house, 2005.

[52] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *Proc. IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[53] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *Proc. ACM MobiSys*, pages 233–245. ACM, 2005.

[54] J. Yang, A. Varshavsky, H. Liu, Y. Chen, and M. Gruteser. Accuracy characterization of cell tower localization. In *Proc. .ACM UbiComp*, pages 223–226. ACM, 2010.

[55] R. Faragher and R. Harle. Location fingerprinting with Bluetooth Low Energy beacons. *IEEE Journal on Selected Areas in Communications*, 33(11):2418–2428, 2015.

[56] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. ACM Mobisys*, pages 15–28. ACM, 2008.

[57] C. Cottrill, F. Pereira, F. Zhao, I. Dias, H. Lim, M. Ben-Akiva, and P. Zegras. Future Mobility Survey: Experience in developing a smartphone-based travel survey in Singapore. *Transportation Research Record: Journal of the Transportation Research Board*, pages 59–67, 2013.

[58] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi. SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *Proc. ACM MobiSys*, 2011.

[59] H. Wang and L.-S. Peh. Mobistreams: A reliable distributed stream processing system for mobile devices. In *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2014.

[60] P. Ji, H.-M. Tsai, C. Wang, and F. Liu. Vehicular visible light communications with led taillight and rolling shutter camera. In *Proc. IEEE VTC*, pages 1–6. IEEE, 2014.

[61] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proc. ACM Sensys*, page 13. ACM, 2013.

[62] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proc. ACM UbiComp*, pages 421–430. ACM, 2012.

[63] X. Zhu, Q. Li, and G. Chen. APT: Accurate outdoor pedestrian tracking with smartphones. In *Proc. IEEE INFOCOM*, pages 2508–2516. IEEE, 2013.

[64] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. Using mobile phone barometer for low-power transportation context detection. In *Proc. ACM Sensys*, pages 191–205. ACM, 2014.

[65] S. Vanini and S. Giordano. Adaptive context-agnostic floor transition detection on smart mobile devices. In *Proc. IEEE Pervasive Computing and Communications Workshops*, pages 2–7. IEEE, 2013.

[66] B. Li, B. Harvey, and T. Gallagher. Using barometers to determine the height for indoor positioning. In *Proc. International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7. IEEE, 2013.

[67] M. Tanigawa, H. Luinge, L. Schipper, and P. Slycke. Drift-free dynamic height sensor using MEMS IMU aided by MEMS pressure sensor. In *Proc. Workshop on Positioning, Navigation and Communication*, pages 191–196. IEEE, 2008.

[68] V. Savov. The LG G3 will be the first smartphone with laser autofocus. `http://www.theverge.com/2014/5/23/5744542/lg-g3-will-be-the-first-smartphone-with-laser-autofocus`, May 2014. (Accessed on 08/15/2016).

[69] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh. Low cost crowd counting using audio tones. In *Proc. ACM Sensys*, pages 155–168. ACM, 2012.

[70] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proc. ACM Sensys*, pages 323–336. ACM, 2008.

[71] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt. Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, 16(3):285–303, 2011.

[72] S. Chen and J. Zhao. The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication. *IEEE Communications Magazine*, 52(5):36–43, May 2014.

[73] R. N. Clarke. Expanding mobile wireless capacity: The challenges presented by technology and economics. *Telecommunications Policy*, 38(8):693–708, 2014.

[74] J. Huang et al. A close examination of performance and power characteristics of 4G LTE networks. In *Proc. ACM MobiSys*, 2012.

[75] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski. Five disruptive technology directions for 5G. *IEEE Communications Magazine*, 52(2):74–80, February 2014.

[76] Cisco. Cisco Visual Networking Index: Global mobile data traffic forecast update, 2015-2020, 2016.

[77] E. Perahia and R. Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. Cambridge University Press, 2013.

[78] Y. Zeng, P. H. Pathak, and P. Mohapatra. A first look at 802.11ac in action: energy efficiency and interference characterization. In *Proc. IFIP Networking Conference*, pages 1–9. IEEE, 2014.

[79] C. Bisdikian et al. An overview of the Bluetooth wireless technology. *IEEE Communications Magazine*, 39(12):86–94, 2001.

[80] L.-J. Chen, T. Sun, and Y.-C. Chen. Improving Bluetooth EDR data throughput using FEC and interleaving. In *Proc. Second International Conference on Mobile Ad-Hoc and Sensor Networks*, pages 724–735. Springer, 2006.

[81] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[82] V. Coskun, B. Ozdenizci, and K. Ok. A survey on near field communication (NFC) technology. *Wireless personal communications*, 71(3):2259–2294, 2013.

[83] Z. Antoniou and S. Varadan. Intuitive mobile user interaction in smart spaces via NFC-enhanced devices. In *Proc. International Conference on Wireless and Mobile Communications (ICWMC)*, pages 86–86. IEEE, 2007.

[84] S. K. Timalsina, R. Bhusal, and S. Moh. NFC and its application to mobile payment: Overview and comparison. In *Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference on*, volume 1, pages 203–206. IEEE, 2012.

[85] Wi-Fi Alliance. Wi-Fi Peer-to-Peer (P2P) technical 7 specification. http://www.wi-fi.org/discover-wi-fi/wi-fi-direct. (Accessed on 08/15/2016).

[86] Apple. Multipeer Connectivity Framework reference. http://developer.apple.com/library/ios/documentation/MultipeerConnectivity/Reference/MultipeerConnectivityFramework/. (Accessed on 08/15/2016).

[87] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. In *Proc. ACM SIGCOMM Conference on Data Communication*, 2008.

[88] Qualcomm. LTE Direct. http://www.qualcomm.com/research/projects/lte-direct, 2010. (Accessed on 08/24/2016).

[89] Google Code. Issue 82 - android - support Wi-Fi ad hoc networking. http://code.google.com/p/android/issues/detail?id=82. (Accessed on 07/31/2016).

[90] Google. Project Soli. http://atap.google.com/soli/. (Accessed on 08/15/2016).

[91] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota. FingerIO: Using active sonar for fine-grained finger tracking. In *Proc. ACM CHI*, pages 1515–1525. ACM, 2016.

[92] S. Greengard. Automotive systems get smarter. *Communications of the ACM*, 58(10):18–20, September 2015.

[93] P. Shelly. Addressing challenges in automotive connectivity: Mobile devices, technologies, and the connected car. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 8(2015-01-0224):161–169, 2015.

[94] S. Elmer. Volkswagen iBeetle unveiled with built-in iPhone dock. http://www.autoguide.com/auto-news/2013/04/volkswagen-ibeetle-unveiled-with-built-in-iphone-dock.html, April 2013. (Accessed on 08/24/2016).

[95] Automatic. Automatic: Connect your car to your digital life. http://www.automatic.com. (Accessed on 08/24/2016).

[96] U.S. Department of Transportation. The connected vehicle test bed. http://www.its.dot.gov/factsheets/connected_vehicle_testbed_factsheet.htm. (Accessed on 08/15/2016).

[97] iFixit. iFixit iPhone 4 teardown. http://ifixit.com/Teardown/iPhone+4/3130, 2010. (Accessed on 08/24/2016).

[98] G. Liu, P. Haldi, T. K. Liu, and A. M. Niknejad. Fully integrated CMOS power amplifier with efficiency enhancement at power back-off. *IEEE Journal of Solid-State Circuits*, 3(43):433–435, March 2008.

[99] P. Choi, J. Gao, N. Ramanathan, M. Mao, S. Xu, C.-C. Boon, S. A. Fahmy, and L.-S. Peh. A case for leveraging 802.11 p for direct phone-to-phone communications. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*. ACM, 2014.

[100] P. Choi, C. Boon, M. Mao, and H. Liu. 28.8 dBm, high efficiency, linear GaN power amplifier with in-phase power combining for IEEE 802.11p applications. *IEEE Microwave and Wireless Components Letters*, 23(8):433–435, August 2013.

[101] J. Lotze, S. A. Fahmy, J. Noguera, B. Ozgul, L. Doyle, and R. Esser. Development framework for implementing FPGA-based cognitive network nodes. In *IEEE Global Communications Conference (GLOBECOM)*, 2009.

[102] M. C. Ng, K. E. Fleming, M. Vutukuru, S. Gross, and H. Balakrishnan. Airblue: A system for cross-layer wireless protocol development. In *Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2010.

[103] T. H. Pham, I. V. McLoughlin, and S. A. Fahmy. Shaping spectral leakage for IEEE 802.11p vehicular communications. In *Proc. IEEE VTC*, 2014.

[104] J. Emer, P. Ahuja, E. Borch, A. Klauser, C.-K. Luk, S. Manne, S. Mukherjee, H. Patil, S. Wallace, N. Binkert, R. Espasa, and T. Juan. Asim: a performance model framework. *Computer*, 35(2):68–76, 2002.

[105] A. Parashar, M. Adler, K. Fleming, M. Pellauer, and J. Emer. LEAP: A virtual platform architecture for FPGAs. In *Proc. Workshop on the Intersections of Computer Architecture and Reconfigurable Logic (CARL)*, 2010.

[106] J. Gao and L.-S. Peh. RoadRunner: Infrastructure-less vehicular congestion control. In *Proc. ITS World Congress*, 2014.

[107] Cohda Wireless. MK2 WAVE-DSRC radio. `http://cohdawireless.com/Products/Hardware.aspx`. (Accessed on 08/24/2016).

[108] Monsoon Solutions. Monsoon Power Monitor. `http://msoon.com/LabEquipment/PowerMonitor/`. (Accessed on 08/24/2016).

[109] D. Schrank and T. Lomax. *The 2007 urban mobility report*. Texas Transportation Institute, Texas A & M University, 2007.

[110] Wikipedia. List of electronic toll collections systems. `http://en.wikipedia.org/wiki/List_of_electronic_toll_collection_systems`. (Accessed on 08/15/2016).

[111] F. T. Seik. An advanced demand management instrument in urban transport: Electronic road pricing in Singapore. *Cities*, 17(1):33–45, 2000.

[112] M. Goh. Congestion management and electronic road pricing in Singapore. *Journal of Transport Geography*, 10(1):29–38, 2002.

[113] C. Keong. Road pricing: Singapore's experience. In *Third seminar of the IMPRINT-EUROPE Thematic Network, "Implementing Reform on Transport Pricing: Constraints and Solutions: Learning from Best Practice"*, 2002.

[114] J. Finn. The rebirth of toll: ETC makes its American comeback. *Traffic Technology International Supplement, Tolltrans supplement*, 2004.

[115] C. Daganzo. A pareto optimum congestion reduction scheme. *Transportation Research Part B: Methodological*, 29(2):139–154, 1995.

[116] K. Nakamura and K. Kockelman. Congestion pricing and roadspace rationing: an application to the San Francisco Bay Bridge corridor. *Transportation Research Part A: Policy and Practice*, 36(5):403–417, 2002.

[117] Wikipedia. Applications of road space rationing. `http://en.wikipedia.org/wiki/Road_space_rationing#Applications_of_road_space_rationing`. (Accessed on 08/15/2016).

[118] S. Phang and R. Toh. From manual to electronic road congestion pricing: The Singapore experience and experiment. *Transportation Research Part E: Logistics and Transportation Review*, 33(2):97–106, 1997.

[119] Y. Zhou, Y. Wu, L. Yang, L. Fu, K. He, S. Wang, J. Hao, J. Chen, and C. Li. The impact of transportation control measures on emission reductions during the 2008 Olympic Games in Beijing, China. *Atmospheric Environment*, 44(3):285–293, 2010.

[120] M. Almenoar. Call for new ERP proposals. *The Straits Times*, June 2010.

[121] Nielsen Co. Smartphones account for half of all mobile phones, 2012.

[122] Nielsen Co. Netizens in Singapore spend more than a day a week on Internet, 2012.

[123] J. Dunn. Cars get connected: The best car tech trends today. `http://www.technologyguide.com/default.asp?newsID=5255`, December 2012. (Accessed on 08/24/2016).

[124] R. Bose, J. Brakensiek, K.-Y. Park, and J. Lester. Morphing smartphones into automotive application platforms. *Computer*, 44(5):53–61, May 2011.

[125] J. Sonnenberg. Service and user interface transfer from nomadic devices to car infotainment systems. In *Proc. ACM AutoUI*, 2010.

[126] Rysavy Research. Mobile broadband capacity constraints and the need for optimization. `http://www.rysavy.com/articles/2010_02_Rysavy_Mobile_Broadband_Capacity_Constraints.pdf`, 2010. (Accessed on 08/24/2016).

[127] E. Koukoumidis et al. Pocket cloudlets. In *Proc. ACM ASPLOS*, 2011.

[128] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8:14–23, 2009.

[129] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty. Performance evaluation of safety applications over DSRC vehicular ad hoc networks. In *Proc. ACM VANET*, 2004.

[130] S. Lu, T. He, and Z. Gao. Electronic toll collection system based on Global Positioning System technology. In *Proc. International Conference on Challenges in Environmental Science and Computer Engineering*, volume 2, March 2010.

[131] W.-H. Lee, B.-S. Jeng, S.-S. Tseng, and C.-H. Wang. Electronic toll collection based on vehicle-positioning system techniques. In *Proc. IEEE ICNSC*, 2004.

[132] D. Srinivasan, R. Cheu, and C. Tan. Development of an improved ERP system using GPS and AI techniques. In *Proc. IEEE ITSC*, 2003.

[133] K. Basak, S. Hetu, Z. Li, C. Lima Azevedo, H. Loganathan, T. Toledo, R. Xu, Y. Xu, L.-S. Peh, and M. E. Ben-Akiva. Modeling reaction time within a traffic simulation model. In *IEEE Intelligent Transportation Systems*, 2013.

[134] I. Leontiadis and C. Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *Proc. IEEE WoWMoM*, 2007.

[135] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. MDDV: a mobility-centric data dissemination algorithm for vehicular networks. In *Proc. ACM VANET*, pages 47–56, 2004.

[136] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[137] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz. Routing in sparse vehicular ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 25(8):1538–1556, 2007.

[138] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Computer Communications*, 31(12):2838–2849, 2008.

[139] B. Parno and A. Perrig. Challenges in securing vehicular networks. In *Proc. ACM HotNets*, 2005.

[140] M. Raya and J. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[141] X. Lin et al. Security in vehicular ad hoc networks. *IEEE Communications Magazine*, 46(4):88–95, 2008.

[142] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proc. ACM IMC*, November 2009.

[143] J. C. Abela and E. M. Aguilar. Panoramio - photos of the world. `http://www.panoramio.com/`. (Accessed on 08/15/2016).

[144] J. Andrus et al. Cells: a virtual mobile smartphone architecture. In *Proc. ACM SOSP*, 2011.

[145] S. Dolev et al. Brief announcement: Virtual stationary automata for mobile networks. In *Proc. ACM PODC*, 2005.

[146] M. Brown et al. The virtual node layer: A programming abstraction for wireless sensor networks. In *Proc. International Workshop on Wireless Sensor Network Architecture (WWSNA)*, 2007.

[147] J. Wu et al. Simulating fixed virtual nodes for adapting wireline protocols to MANET. In *Proc. IEEE NCA*, 2009.

[148] L. Breslau, D. Estrin, H. Yu, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, et al. Advances in network simulation. *Computer*, 33(5):59–67, May 2000.

[149] K. Gharachorloo et al. Memory consistency and event ordering in scalable shared-memory multiprocessors. In *Proc. ISCA*, 1990.

[150] S. V. Adve and K. Gharachorloo. Shared memory consistency models: A tutorial. *Computer*, 29:66–76, December 1996.

[151] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM SIGOPS Operating Systems Review*, 17(5):3, 1983.

[152] J. R. Goodman. Using cache memory to reduce processor-memory traffic. In *Proc. ISCA*, pages 124–131, 1983.

[153] D. Lenoski et al. The directory-based cache coherence protocol for the DASH multiprocessor. In *Proc. ISCA*, 1990.

[154] M. M. K. Martin et al. Timestamp snooping: an approach for extending SMPs. In *Proc. ACM ASPLOS*, pages 25–36, New York, NY, USA, 2000. ACM.

[155] N. Agarwal, L.-S. Peh, and N. K. Jha. In-Network Snoop Ordering (INSO): Snoopy coherence on unordered interconnects. In *Proc. IEEE HPCA*, 2009.

[156] J. Gettys and K. Nichols. Bufferbloat: Dark buffers in the internet. *ACM Queue*, 9(11):40:40–40:54, November 2011.

[157] A. Rahmati and L. Zhong. Context-for-wireless: Context-sensitive energy-efficient wireless data transfer. In *Proc. Mobisys*, 2007.

[158] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *Proc. IEEE VTC*, pages 2036–2040. IEEE, 2008.

[159] R. O'Toole. A free parking space grows in Manhattan. `http://ti.org/antiplanner/?p=3565`, 2010. (Accessed on 08/24/2016).

[160] P. Sewell et al. x86-TSO: a rigorous and usable programmer's model for x86 multiprocessors. *Communications of the ACM*, 53, July 2010.

[161] IBM. *IBM System/370 Principles of Operation.* IBM, May 1983.

[162] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *Transactions on Computers*, 28:690–691, September 1979.

[163] Alpha Architecture Committee. *Alpha Architecture Reference Manual.* Digital Press, 1992.

[164] SPARC International. The SPARC architecture manual, 1994.

[165] M. Herlihy and J. E. B. Moss. Transactional memory: architectural support for lock-free data structures. In *Proc. ISCA*, pages 289–300, New York, NY, USA, 1993. ACM.

[166] R. Gummadi et al. Kairos: A macro-programming system for wireless sensor networks. In *Proc. ACM SOSP*, 2005.

[167] N. Kothari et al. Reliable and efficient programming abstractions for wireless sensor networks. In *Proc. ACM SIGPLAN PLDI*, 2007.

[168] S. Dolev et al. GeoQuorums: Implementing atomic memory in mobile ad hoc networks. *Distributed Computing*, 18(2):125–155, 2005.

[169] G. Chockler, S. Gilbert, and N. Lynch. Virtual infrastructure for collision-prone wireless networks. In *Proc. ACM PODC*, 2008.

[170] D. B. Terry et al. Managing update conflicts in bayou, a weakly connected replicated storage system. In *Proc. ACM SOSP*, 1995.

[171] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda File System. *ACM Transactions on Computer Systems*, 10, February 1992.

[172] D. Chen et al. InterWeave: A middleware system for distributed shared state. In *Proc. International Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers*, 2000.

[173] C. Amza et al. TreadMarks: Shared memory computing on networks of workstations. *Computer*, 29, February 1996.

[174] R. W. Wolcott and R. M. Eustice. Visual localization within LIDAR maps for automated urban driving. In *Proc. IEEE IROS*, pages 176–183. IEEE, 2014.

[175] D. Kohler and K. Conley. rosjava–an implementation of ROS in pure Java with Android support. `http://github.com/rosjava/rosjava_core`, 2011. (Accessed on 08/24/2016).

[176] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah. A low-cost laser distance sensor. In *Proc. IEEE ICRA*, pages 3002–3008. IEEE, 2008.

[177] A. Colaco, A. Kirmani, N.-W. Gong, T. McGarry, L. Watkins, and V. K. Goyal. 3dim: Compact and low power time-of-flight sensor for 3D capture using parametric signal processing. In *Proc. International Image Sensor Workshop*, pages 349–352, 2013.

[178] K. Kimoto, N. Asada, T. Mori, Y. Hara, A. Ohya, and S. Yuta. Development of small size 3D LIDAR. In *Proc. IEEE ICRA*, pages 4620–4626. IEEE, 2014.

[179] M. J. Cŕee, L. V. Streeter, R. M. Conroy, and A. A. Dorrington. Analysis of the SoftKinetic DepthSense for range imaging. In *Image Analysis and Recognition*, pages 668–675. Springer, 2013.

[180] C. S. Bamji, P. O'Connor, T. Elkhatib, S. Mehta, B. Thompson, L. A. Prather, D. Snow, O. C. Akkaya, A. Daniel, A. D. Payne, T. Perry, M. Fenton, and V. H. Chan. A 0.13 um CMOS System-on-Chip for a 512 x 424 time-of-flight image sensor with multi-frequency photo-demodulation up to 130 MHz and 2 GS/s ADC. *IEEE Journal of Solid-State Circuits*, 50(1):303–319, Jan 2015.

[181] K. Khoshelham and S. O. Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[182] Intel. Get started with RealSense technology. `http://software.intel.com/en-us/realsense/devkit`. (Accessed on 08/24/2016).

[183] Intel. Outside/sunlight functionality and Linux support. `http://software.intel.com/en-us/forums/realsense/topic/532486`. (Accessed on 08/24/2016).

[184] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. V. Le, A. Wellman, and A. Y. Ng. High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening. In *Proc. IEEE ICRA*, pages 2816–2822, 2009.

[185] Office for Research Safety. *Laser safety handbook.* Northwestern University, 2011.

[186] A. K. Maurya and P. S. Bokare. Study of deceleration behaviour of different vehicle types. *International Journal for Traffic and Transport Engineering*, 2(3):253–270, 2012.

[187] OmniVision Technologies. OmniVision's global shutter CameraCubeChip brings computer vision to mobile devices, notebooks and wearables. `http://www.prnewswire.com/news-releases/277202061.html`, 2014. (Accessed on 08/24/2016).

[188] D. Moloney and O. D. Suarez. A vision for the future [soapbox]. *IEEE Consumer Electronics Magazine*, 4(2):40–45, 2015.

[189] Primate Labs. LG Nexus 5 vs HTC One M9. `http://browser.primatelabs.com/geekbench3/compare/583107?baseline=2326098`, 2014. (Accessed on 08/24/2016).

[190] M. Harris. Researcher hacks self-driving car sensors. `http://spectrum.ieee.org/cars-that-think/transportation/self-driving/researcher-hacks-selfdriving-car-sensors`, 2015. (Accessed on 08/24/2016).

[191] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky. Virtual individual servers as privacy-preserving proxies for mobile devices. In *Proc. ACM SIGCOMM MobiHeld*, MobiHeld '09, pages 37–42, New York, NY, USA, 2009. ACM.

[192] S. Jaiswal and A. Nandi. Trust no one: A decentralized matching service for privacy in location based services. In *Proc. ACM SIGCOMM MobiHeld*, MobiHeld '10, pages 51–56, New York, NY, USA, 2010. ACM.

[193] J. Winter. Trusted computing building blocks for embedded Linux-based ARM TrustZone platforms. In *Proc. ACM STC*, 2008.

[194] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory air pollution monitoring using smartphones. *Mobile Sensing*, pages 1–5, 2012.

[195] Y. Ishigaki, Y. Matsumoto, R. Ichimiya, and K. Tanaka. Development of mobile radiation monitoring system utilizing smartphone and its field tests in Fukushima. *IEEE Sensors Journal*, 13(10):3520–3526, 2013.

[196] H. Kailanto, E. Hyvarinen, and J. Hyttinen. Mobile ECG measurement and analysis system using mobile phone as the base station. In *Proc. EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 12–14. IEEE, 2008.

[197] P. Aspinall, P. Mavros, R. Coyne, and J. Roe. The urban brain: analysing outdoor physical activity with mobile EEG. *British Journal of Sports Medicine*, pages bjsports–2012, 2013.

[198] H. McCracken. Project Ara: Inside Google's bold gambit to make smartphones modular. *TIME Magazine*, 5, 2014.

[199] LG Electronics. LG opens up LG G5 modular design to developers. http://www.prnewswire.com/news-releases/lg-opens-up-lg-g5-modular-design-to-developers-300249876.html, April 2016. (Accessed on 08/15/2016).

[200] Verizon. Moto Z Droids and Moto Mods are now available for preorder exclusively on Verizon. http://www.prnewswire.com/news-releases/moto-z-droids-and-moto-mods-are-now-available-for-pre-order-exclusively-on-verizon-300302000.html, July 2016. (Accessed on 08/15/2016).

[201] Gartner. Gartner says 4.9 billion connected "things" will be in use in 2015. http://www.gartner.com/newsroom/id/2905717, November 2014. (Accessed on 08/15/2016).

[202] Gartner. Gartner says worldwide wearable devices sales to grow 18.4 percent in 2016. http://www.gartner.com/newsroom/id/3198018, February 2016. (Accessed on 08/15/2016).

[203] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar. An early resource characterization of deep learning on wearables, smartphones and Internet-of-Things devices. In *Proc. International Workshop on Internet of Things Towards Applications*, IoT-App '15, pages 7–12, New York, NY, USA, 2015. ACM.

[204] C. Tan, K. Aditi, V. Venkataramani, M. Karunaratne, T. Mitra, and L.-S. Peh. LOCUS: Low-power customizable many-core architecture for wearables. In *Proc. International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. IEEE, 2016.