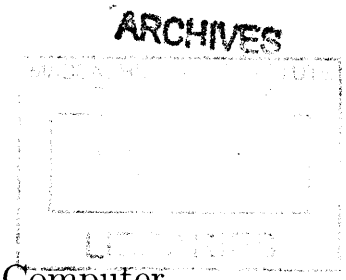
Distributed Mobile Platforms and Applications for Intelligent Transportation Systems



by

Jason Hao Gao

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

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2013

Author .....
Department of Electrical Engineering and Computer Science
January 18, 2013

Certified by .....
Li-Shiuan Peh
Associate Professor
Thesis Supervisor

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

Distributed Mobile Platforms and Applications for Intelligent Transportation Systems

by

Jason Hao Gao

Submitted to the Department of Electrical Engineering and Computer Science
on January 18, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Smartphones are pervasive, and possess powerful processors, multi-faceted sensing, and multiple radios. However, 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 local compute power and device-to-device communications of modern smartphones can mitigate demand on cellular networks and improve response times. This thesis presents two systems towards this vision.

First, we present DIPLOMA, which aids developers in achieving this vision by providing a programming layer to easily program a collection of smartphones connected over adhoc wireless. It presents a familiar shared data model to developers, while underneath, it implements a distributed shared memory system that provides coherent relaxed-consistency access to data across different smartphones and addresses the issues that device mobility and unreliable networking pose against consistency and coherence. We evaluated our prototype on 10 Android phones on both 3G (HSPA) and 4G (LTE) networks with a representative location-based photo-sharing service and a synthetic benchmark. We also simulated large scale scenarios up to 160 nodes on the ns-2 network simulator. Compared to a client-server baseline, our system shows response time improvements of 10x over 3G and 2x over 4G. We also observe cellular bandwidth reductions of 96%, comparable energy consumption, and a 95.3% request completion rate with coherent caching.

With RoadRunner, we apply our vision to Intelligent Transportation Systems (ITS). RoadRunner implements vehicular congestion control as an in-vehicle smartphone app that judiciously harnesses onboard sensing, local computation, and short-range communications, enabling large-scale traffic congestion control without the need for physical infrastructure, at higher penetration across road networks, and at finer granularity. RoadRunner enforces a quota on the number of cars on a road by requiring vehicles to possess a token for entry. Tokens are circulated and reused among multiple vehicles as they move between regions. We implemented RoadRunner as an Android application, deployed it on 10 vehicles using 4G (LTE), 802.11p DSRC

and 802.11n adhoc WiFi, and measured cellular access reductions up to 84%, response time improvements up to 80%, and effectiveness of the system in enforcing congestion control policies. We also simulated large-scale scenarios using actual traffic loop-detector counts from Singapore.

Thesis Supervisor: Li-Shiuan Peh
Title: Associate Professor

Acknowledgments

I give my sincerest gratitude to the following people:

My advisor, Professor Li-Shiuan Peh, for her seemingly limitless energy and optimism, immeasurable helpfulness in research guidance, and unwavering eagerness to participate in experiments.

My academic advisor, Professor Randall Davis, who patiently guided me through departmental requirements and petitions, and profferred valuable insight on the research process.

Anirudh Sivaraman, who has provided fruitful late-night discussion, shell scripting tips, humorous banter, and last-minute salvation on uncountable occasions.

Niket Agarwal and HaoQi Li, who, along with Anirudh, were my collaborators on the DIPLOMA project, and the many volunteers who participated in the experiments for DIPLOMA.

My lab-mates Tushar, Owen, Bhavya, Sunghyun, Huayong, Woo-Cheol, Suvinay, Pablo, Kostas, and Manos for their invaluable advice concerning graduate school and research matters, help in conducting experiments, and convivial conversation in 32-G785.

Frances, who supported me and brought dinner and snacks when I was stuck late in lab; my friends, who have given me light-hearted respite; and finally, my family, who have unconditionally supported my aspirations and endeavors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Smart mobile devices are increasingly pervasive and powerful: a modern smartphone has several communications interfaces, multi-faceted sensing capabilities, and multiple processing cores, all in a single device:

1. **Cellular radio:** 3G cellular data is prevalent in many parts of the world, and 4G technologies such as LTE and WiMax are already deployed in many cities.

2. **Device-to-device communications:** A smartphone contains multiple short-range wireless interfaces including WiFi, Bluetooth, and NFC.

3. **Local computation:** Modern smartphones are also quite powerful, commonly possessing quad-core processors such as the Nvidia Tegra 3 [58], Samsung Exynos Quad [68], and Qualcomm S4 Pro [64].

4. **Multi-faceted sensing:** Smartphones have many sensors, including GPS, accelerometer, gyroscope, compass, barometer, multiple microphones, ambient light, and proximity.

Furthermore, programmers can readily write mobile applications which will run on millions of smartphones using popular platforms such as Google Android and Apple iOS. The functions of popular mobile applications range from email, chat, social networking, games, and photo sharing to mobile payments, taxi bookings, location-based services, review aggregation, and video streaming.

These networked mobile apps typically use a client-server model, where a thin front-end application on the mobile devices primarily presents a graphical interface and content and information to the user, while relying on a back-end server to store, retrieve, and run computations on data. Many of these apps access data on a remote server over the cellular network. Writing these applications with a client-server model simplifies system design, but the heavy dependence on the cellular data network has several disadvantages, including the consumption of limited cellular data allocations, use of power-hungry cellular radios, variable and high communications latencies, and limited cellular bandwidth (See Section 2.1).

If networked mobile apps could more fully leverage the local computation, sensing, and communications capabilities of modern mobile devices, we can reduce bandwidth pressure on already overloaded cellular networks and improve application responsiveness. This is the direction explored in this thesis. To realize such a vision, an alternative programming model to the client-server model is first needed. DIPLOMA addresses this, proposing and prototyping an alternative distributed mobile programming model. ROADRUNNER seeks to illustrate how networking many mobile phones together can lead to an effective computing platform for infrastructure-less Intelligent Transportation Systems (ITS) services.

1.1 DIPLOMA

In Chapter 2, we present the design, implementation, and evaluation of DIPLOMA [26], a system enables mobile app developers to easily program a collection of mobile devices as if they were using a familiar shared memory model. DIPLOMA provides coherent relaxed-consistency shared memory across different mobile devices in a geographic.

DIPLOMA is a platform upon which mobile application developers can take advantage of device-to-device communications and sensing to offload cellular data usage, without having to design and implement a distributed system themselves. DIPLOMA manages and abstracts away difficult issues such as device mobility, unreliable wire-

less communications, cache coherence, allowing mobile application developers to write applications using a familiar shared memory model.

1.2 RoadRunner

In Chapter 3, we present the design, implementation, and evaluation of RoadRunner, a system that provides infrastructure-less congestion control for vehicular traffic.

Traffic congestion in urban areas is a widespread problem, causing delays and lost productivity. Current congestion control schemes require the deployment of physical infrastructure such as tollbooths, gantries and specialized in-vehicle units, resulting in high cost and logistical difficulty of implementation at scale. Vehicles are becoming increasingly intelligent, however, with built-in computing, communications and integrated smartphone docks.

RoadRunner harnesses device-to-device communications over DSRC or WiFi radios, local vehicle positioning through GPS, and a distributed token reservation protocol to enable highly granular, high-penetration deployment of vehicular congestion control at large scale by using a mobile phone inside each vehicle. RoadRunner eliminates the need to build costly physical infrastructure such as tollbooths and gantries, and judiciously utilizes DSRC or WiFi radios when possible to reduce pressure and demand on cellular networks.

With these two systems, we demonstrate the viability and promise of leveraging sensing, local computation, and device-to-device communications to enable an alternative to the traditional client-server model for mobile applications.

Chapter 2

DIPLOMA: Consistent and Coherent Shared Memory over Mobile Phones

This chapter contains joint work with Anirudh Sivaraman, HaoQi Li, and Niket Agarwal.

2.1 Introduction

Mobile devices are now ubiquitous. Equipped with sophisticated sensors such as GPS, camera, accelerometer and more, they already sense and generate large amounts of data. With quad-core [68] phones now on the market, smartphones will increasingly be able to compute on 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 access networks:** Wireless spectrum is at a premium [44]. Next-generation cellular data networks (4G/LTE) are unlikely to fix this for two reasons: 1) 4G networks are now a substitute for home broadband;

2) Higher screen resolutions are increasing user demand for high bandwidth content such as streaming video. A recent study projected demand to exceed capacity on cellular networks by 2014 [67].

2. **Long and variable latencies:** Cellular networks are characterized by long and highly variable latencies, degrading application response times [41, 70]. Our own measurements in Section 2.4 confirm that 3G latencies can be as high as 50 seconds. 4G latencies are currently significantly better (Section 2.4), but performance on 4G networks will also degrade as user adoption increases.

3. **Poor battery life:** Cellular data transmission drains energy [9], a primary resource for mobile phones.

4. **Monetary cost:** Cellular service plans are increasingly metered and monthly caps are common [44].

We propose moving to a shared memory programming model for location-based services, addressing the issues above by leveraging free, energy-efficient, and low-latency adhoc WiFi to replace cellular accesses 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. Thus, we make the following contributions in this paper:

1. We design and implement DIPLOMA, a Distributed Programming Layer Over Mobile Agents, enabling distributed programming by exposing a shared memory model to the application developer (Sections 2.2 and 2.3).

2. We implement an app similar to the popular location-based photo sharing service on Google Maps, Panoramio [1], and a synthetic benchmark, and measured substantial benefits in latency and cellular bandwidth reduction compared to a conventional client-server implementation on 3G and 4G (Section 2.4).

2.2 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[1] 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 Core (VCore) with its own memory (Section 2.2.1). To address the unreliability of the wireless interconnect, we relax our memory consistency model (Section 2.2.2). Additionally, we cache to speed up remote reads, and propose Snoopy, Resilient Cache Coherence (SRCC) to maintain coherence (Section 2.2.3).

2.2.1 The Virtual Core layer (VCore)

VCores provide the abstraction of static reliable cores interconnected via a 2D mesh. We leverage Virtual Nodes(VN) [21], which abstracts a collection of unreliable mobile nodes in direct communication range of each other[2] into a stationary reliable virtual node. In the original VN system [13], 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

[1]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 [7].

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

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 [13], on a small set of PDAs. Another implementation [88] simulates VNs on the ns-2 [24] 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.

2.2.2 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[3] 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) [28]. We

[3]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

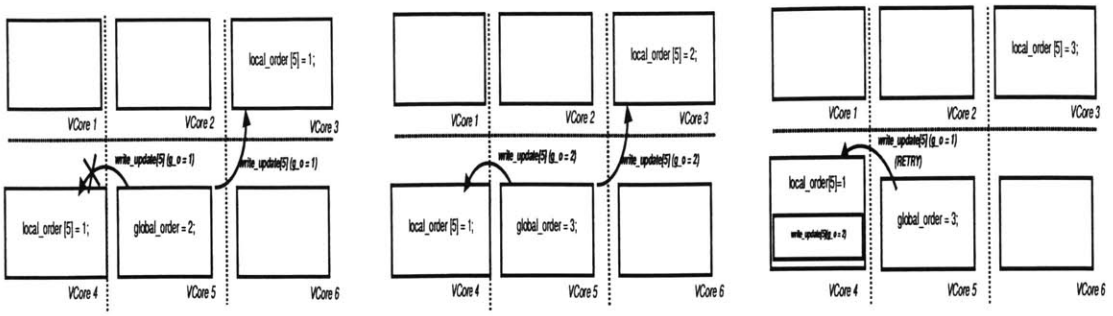
discuss similarities and differences with RC in detail in Section 2.5. We guarantee relaxed consistency [2] by default and allow Atoms to be reordered by the unreliable wireless network. To optionally enforce stricter ordering between atoms, we provide *AtomFence*, a per-*home* VCore memory fence primitive 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* [10] 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.

2.2.3 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 [30] or directory-based [47]. 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 [53] and INSO [3], 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



(a) State of system when first write update of VCore 5 is broadcast. The update does not reach VCore 4.

(b) State of system when second write update of VCore 5 is broadcast. This update reaches both VCores 4 and 3.

(c) First write update of VCore 5 retried. Second write update is buffered at VCore 4 with a *global_order* of 2.

Figure 2-1: Walkthrough example of SRCC for two writes to VCore 5. Only VCores 3, 4 ,5 are detailed for clarity.

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 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 2-1 walks through one such transaction of SRCC.

2.3 DIPLOMA Implementation

2.3.1 DIPLOMA's API

Table 2.1 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 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.

2.3.2 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.

Table 2.1: 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.

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.

2.3.3 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 2.4.1) 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-

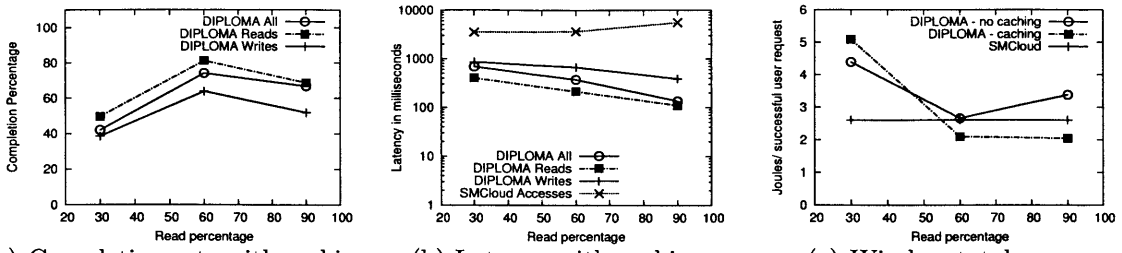
out, and initiates leadership hand-off. It also monitors for unexpected node failures with a leader-to-cloud heartbeat (every 120 seconds in our implementation), so that the server will become aware of node failures and allow a new node to become the leader with the last known state.

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.

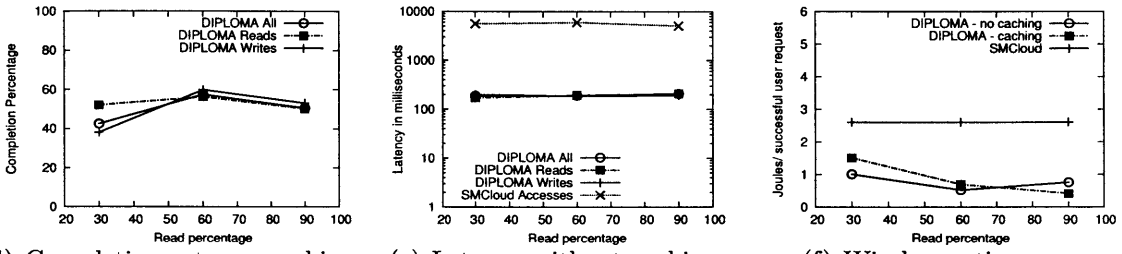
Intermittent cellular connectivity. When a node needs to make a cellular access, e.g. upon entering an empty region, it sends a request to the server to become the VCore, retrying if the server is unreachable. Thus, for DIPLOMA to work, the cellular connection must be eventually available. Current metropolitan cellular networks exhibit this behavior; in our benchmark deployment (Subsection 2.4.1), 3G was available 98% of the time.

2.4 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 [1] 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.



(a) Completion rate with caching (b) Latency with caching (c) Wireless total power



(d) Completion rate, no caching (e) Latency without caching (f) Wireless active power

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

2.4.1 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 2-2(b) and

2-2(e)[4]. 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 2-2(a) and 2-2(d)). Again, we measure reads and writes separately and in aggregate, and compare the completion rate of DIPLOMA to SMCloud.

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

[4]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.

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 [27]. We discuss DIPLOMA's completion rate further in Section 2.4.3.

Power consumption: We use the Monsoon power meter [54] 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 [9, 65], 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 [9, 65], 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[5]. The 3G radio does not return to a low

[5]2 seconds being a realistic time between user interactions. We choose not to batch requests since

power state between requests in SMCloud due to cloud accesses being much more frequent; therefore, measurements include 3G tail energy [9] 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 2-2(f), 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 2-2(c)), 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 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 2-3 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.

---

they are user-initiated, and to maintain a responsive user experience, should not be delayed

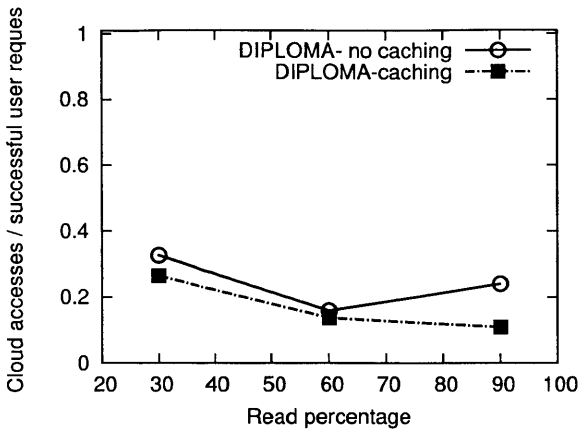


Figure 2-3: Number of cloud accesses

2.4.2 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 2.2 and 2.3, omitting

Table 2.2: 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 2.3: 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

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.

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.

2.4.3 Simulation studies

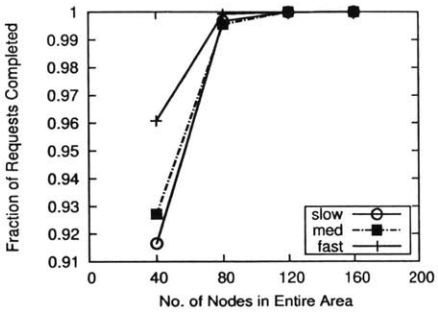
We use ns-2.37 [24], a discrete event network simulator, to evaluate our system at scale with the synthetic benchmark. Node mobility is simulated with the Random Way Point model with three settings: slow, medium and fast (Figure 2.4). Node movements are constrained to a $350m \times 350m$ terrain and the radio range is fixed at 250m. 250m is well within the transmission range of 802.11p or DSRC [36], 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 [59], or about 80-380 cars for our $350m \times 350m$ terrain. Figure 2-4(a) shows the effect of varying the number of nodes on the completion 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%.

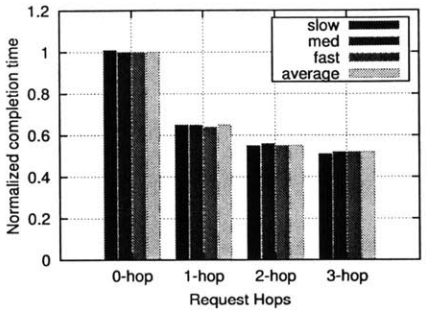
Usefulness of caching. One intuitively expects caching to be more useful for reads to farther away regions. Writes would also take longer since they trigger updates in SRCC. To study this, in Figure 2-4(b) we plot the completion time of a request with caching enabled for varying node speeds. The numbers 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

Table 2.4: 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 2-4: Completion rate and latency of DIPLOMA in simulation.

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.

2.5 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 [75] and

IBM PC [34], stay close to sequential consistency [43] by reordering only certain instruction combinations. Similar to DIPLOMA, some processor architectures (Alpha [5], Sparc [77]) 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) [28]. RC defines memory operations as either *ordinary* or *special*. *Special* operations are either synchronization or non-synchronization accesses. Synchronizing accesses are either *acquires* or *releases*. 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 [32]: 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 [31], an extension of Python, abstracts a sensor network as a collection of nodes which can be tasked simultaneously within a single program. Pleiades [40] 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) [21] is one such abstraction. Section 2.2.1 discussed the practical issues with VN. Geoquorums [22] provides consistency support using a quorum-based algorithm to construct consistent atomic memory over VNs, but it assumes reliable physical layer communication. [16] 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 [80] 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 [39], 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 [15] 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, TreadMarks [6] 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 3

RoadRunner: Infrastructure-less Vehicular Congestion Control

3.1 Introduction

Traffic congestion is a widespread problem affecting road transportation infrastructure in many cities, and is expected to increase in severity [78]. In 2005, congestion resulted in 4.2 billion hours of travel delay and 2.9 billion gallons of wasted fuel in the United States [73]. One widely studied approach to reducing congestion is road pricing [50], 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 widespread in many cites [84]. Here, we discuss several systems along with their implementation:

1. **Singapore.** The Electronic Road Pricing (ERP) system deployed 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 [74, 29, 38].

2. **London.** The London Congestion Charging Scheme (LCCS) [69] installed in

2003 charges a fixed per-day price for vehicles entering a controlled area and uses CCTV cameras mounted atop poles to automatically detect car license plates. Nightly, detected plates are matched with a database for accounting, and missed vehicles are manually verified.

3. **United States.** 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) [25]. 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, also known as road-space rationing. Daganzo [20] proposed a pareto optimum congestion reduction scheme that, in certain cases, can improve everyone's utility, through a hybrid road pricing and road-space rationing strategy, and Nakamura et al [55] have also shown that pure road-space rationing can perform better than any combination of road pricing and road-space rationing. In short, prior studies have shown that road-space rationing schemes are a valuable tool in a transportation engineer's arsenal. Road-space rationing has past and current deployments in several cities around the world [83], mostly with simple, manual, policy-driven implementations:

1. **Singapore's Area Licensing Scheme** [62] prior to the ERP system required vehicles to purchase and display a paper license before entering a restricted zone (RZ). The number of licenses was a fixed quota and ALS was manually enforced by officers at the boundaries of the RZ. 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 priced based on auctioning, with an average price of S$70K-90KCOE for December 2012.

2. **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 successful three-month trial which improved urban traffic conditions [90]. London similarly enforced a road-space rationing scheme for the 2012 Olympics.

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. As a result, deployment of congestion control tends to be limited to few selected regions within cities, with regions covering a wide swath of roads. It is hence very costly to re-define controlled 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 [4], with field trials currently underway.

In this paper, we propose, design and deploy RoadRunner, an infrastructure-less congestion control system that simply runs on ubiquitous smartphones. Today, smartphones are widely adopted in most cities, with penetration reaching 50.4% [57] and 70% [56] in the U.S. and Singapore respectively. Phones can be readily plugged in vehicles, with car manufacturers providing docks for smartphones onto the dashboard [23], enabling seamless connectivity to a substantial energy source, driver-friendly interfaces, vehicular-context information as well as vehicular communications such as DSRC [11, 76][1] Using smartphones enables an already widespread infrastructure-less solution to congestion control, but comes with additional challenges:

1. **Inaccuracies in localization.** Physical infrastructure enables precise detec-

[1]All vehicles in Singapore are required to install an in-vehicle unit equipped with DSRC for communications with ERP gantries. These units can potentially be leveraged for pervasive V2V communications if modified to support the 802.11p DSRC standard.

tion of entries and exits of road regions. The GPS modules on smartphones, however, are often optimized for low power at the expense of reduced sensitivity.

2. **High cellular bandwidth presure.** As smartphones and other mobile connected devices continue to proliferate, demand for cellular bandwidth is expected to exceed available capacity by 2014 [67]. 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 [44]. 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 networks.

3. **Low response latency.** Phone-based congestion control needs to swiftly respond to drivers so they can adapt their routing appropriately. A conventional client-server phone app implementation which relies on the cellular data network may experience long and unpredictable latencies, especially when the network is heavily loaded in a dense region [42, 71], and face difficulties meeting the real-time requirements of congestion control.

RoadRunner tackles the above challenges by judiciously leveraging the myriad networking interfaces, sensing and substantial computing capability of state-of-the-art smartphones 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 bandwidth pressure on 4G/LTE 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. For localization, our experiments demonstrate that today's smartphone GPS receivers have high accuracy and sensitivity, that, when combined with buffer zones between regions at

intersections, enable accurate identification of controlled region entries and exits.

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 simulation results (Section 3.5) indicate that such an approach can enable infrastructure-less congestion control on a large scale at realistic vehicle densities.

3.2 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 where vehicles must possess a corresponding *token* to drive on a specific road segment within an enforced region that is pre-defined, which is analagous 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.

We conducted micro-experiments to answer key design questions and guide our final design:

- Can we accurately and quickly enforce boundaries of pre-defined regions in an infrastructure-less manner, using the GPS receivers on commodity smartphones?
- Can vehicle-to-vehicle (V2V) communications offload cellular bandwidth pressure and lead to better response times?

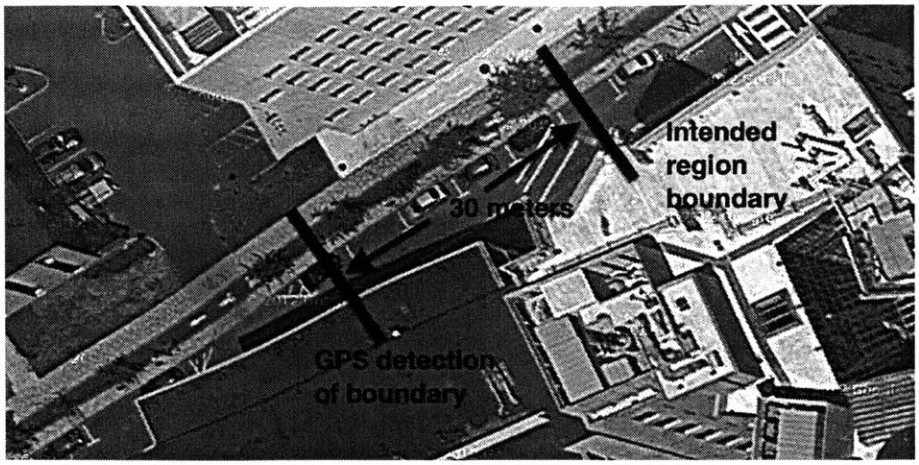


Figure 3-1: Map of programmed region boundary and detected region boundary for static GPS microexperiment.

3.2.1 Boundary enforcement microexperiments

We implemented a prototype of RoadRunner using the built-in GPS receiver on Samsung Galaxy Note smartphones (See Section 3.3 for details) and conducted experiments with a vehicle driving on Vassar Street in Cambridge, Massachusetts, USA. The vehicle crossed a region boundary at vehicular speeds of 15-20 mph. We predefined the region boundary as a transportation engineer would in our system by calculating the coordinates of the region boundary from Google Maps satellite images and programming the coordinates into our RoadRunner implementation.

Boundary detection accuracy

We first measured the accuracy of the phone's GPS when held statically by a person standing on a street (Vassar Street, Cambridge, MA). We noted where the phone detected the boundary crossing by walking a bit down the street, stopping, and noting whether the phone detected a boundary crossing. When the phone detected the boundary crossing, we honed in on it by walking back and forth across it in successively smaller distances. We visually referenced this location to satellite images, and measured the distance of this detected boundary from where we expected the programmed boundary to be. This resulted in a boundary detection inaccuracy of 30 meters, as shown in Figure 3-1.

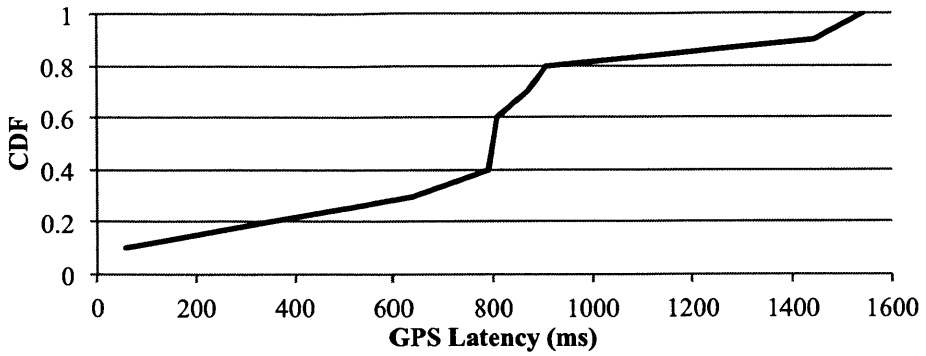


Figure 3-2: CDF of boundary detection latency, 10 trials.

Boundary detection latency

Next, we conducted 10 trials of driving across the region boundary at speeds of 15-20 mph on Vassar Street, in a car with RoadRunner installed. The driver pressed a button to indicate when he crossed the region boundary, producing a timestamp to compare to the one created when the system detects a boundary crossing. The boundary detection latency is the time between physically crossing a detected region boundary and when the system reports a boundary crossing. This delay averaged 821 milliseconds, with a maximum detection latency of 1.5 seconds (Figure 3-2), sufficient to detect a vehicle within 47 meters even at high travel speeds of 70 mph. These results indicate that transportation authorities should define controlled regions that are sufficiently long to ensure that vehicles do not pass undetected through a region.

These experiments gave us confidence that the integrated GPS receiver on a smartphone delivers sufficient accuracy and responsiveness in boundary enforcement for RoadRunner. Ensuring a minimum buffer distance between region boundaries suffices to enable correctly detected boundary crossings. The road intersections in our Cambridge, MA, USA deployment (Section 3.4) are 30 meters across, and we place region boundaries at least 20 meters before the intersections, producing buffer distances between boundaries of at least 80 meters.

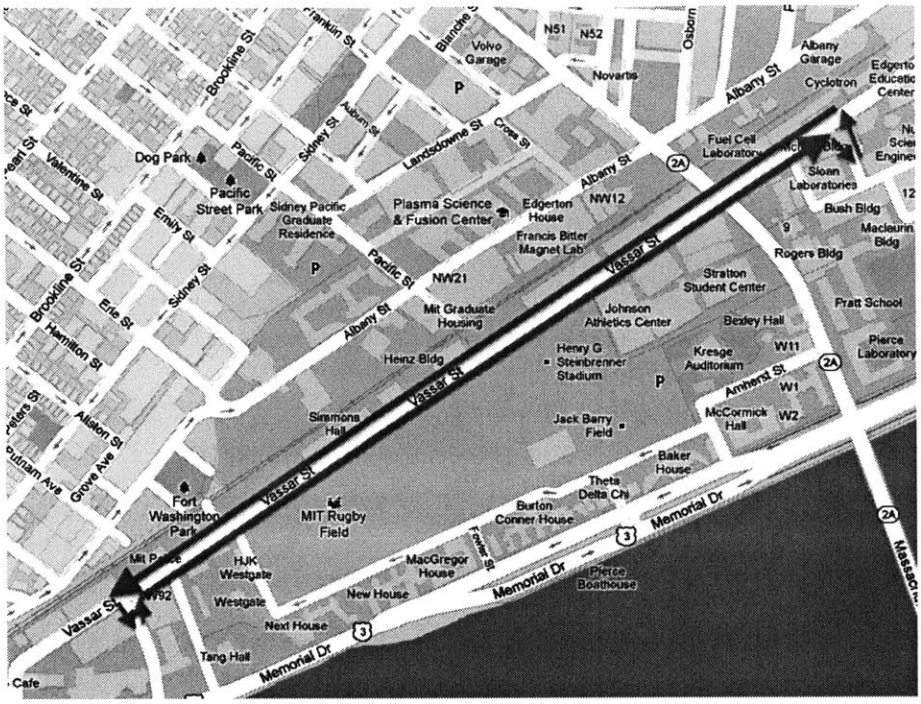


Figure 3-3: V2V microexperiment test route

3.2.2 Vehicle-to-vehicle and vehicle-to-cloud microexperiments

We compared Vehicle-to-Vehicle (V2V) and Vehicle-to-Cloud (V2Cloud) communications response latencies for requests occuring Vehicle-to-Cloud over 4G LTE cellular data (V2Cloud), Vehicle-to-Vehicle over adhoc Wi-Fi (V2V-WiFi), and Vehicle-to-Vehicle over DSRC (V2V-DSRC). All interactions are timestamped on the phone from the instant a request is sent to when the corresponding response is received to obtain end-to-end system latencies.

We benchmarked V2V communications in 2-car microexperiments. The vehicles looped back and forth on a straight 1.2 kilometer segment of Vassar Street (Figure 3-3) in an urban setting at speeds varying from 0 to 30 mph, in various situations including one car following another, one car stationary, both cars stationary, and both cars driving in opposite directions and passing by each other. The V2V-DSRC and V2V-WiFi experiments occurred simultaneously over 1.5 hours, and the cars continuously exchanged a token back and forth when they were within communications range.

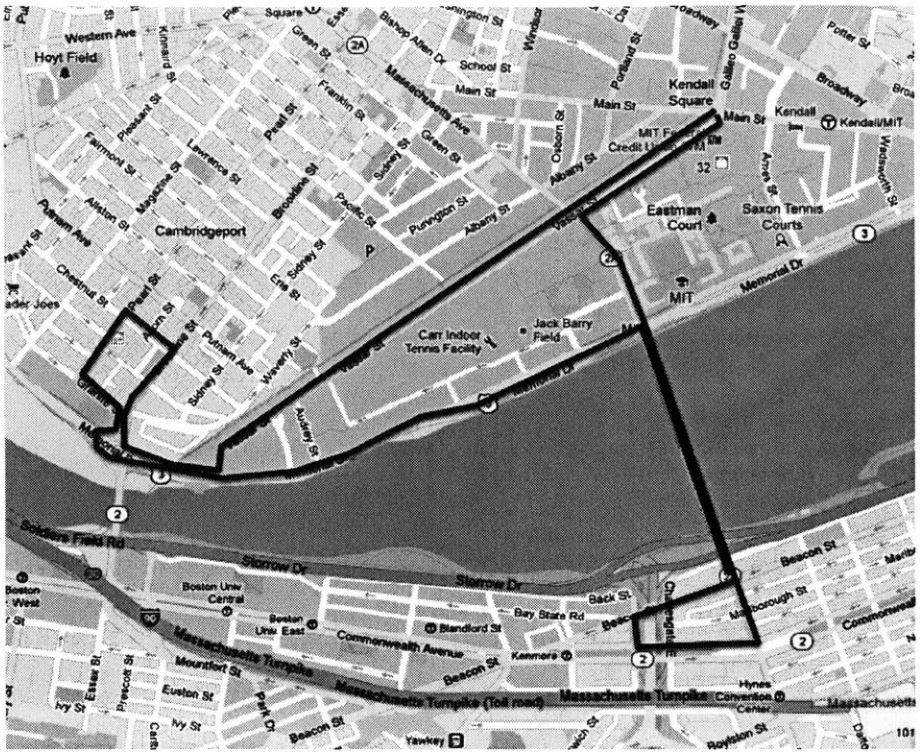


Figure 3-4: V2Cloud microexperiment test route

The V2Cloud measurements occurred over 20 minutes of a single vehicle driving through Cambridge and Boston on the urban route in Figure 3-4, with speeds varying from 0 to 30 mph. We drove a longer and more diverse route for this experiment as we did not have to coordinate the movements of two cars as in the V2V experiments. The system exchanged a token between the vehicle and the server every 2 seconds.

These response latency microexperiments generated 13067 V2V-DSRC, 1375 V2V-WiFi, and 473 V2Cloud end-to-end latency measurements (Figure 3-5). There are many more V2V-DSRC measurements than V2V-WiFi because the DSRC radios are able to communicate and exchange tokens at longer distances than the WiFi radios: in the scenario with two vehicles driving in opposing directions, with inter-vehicle distance varying from 1.2 km to 0 m, we observed that the distance of token exchanges were 7.4 meters mean and 6.8 meters median over 45 WiFi exchanges, vs. 100.2 meters mean and 99.6 meters median over 519 DSRC exchanges. It should be noted that these experiments were carried out largely to test the radios and obtain initial numbers for guiding the system design, and not intended as a rigorous head-

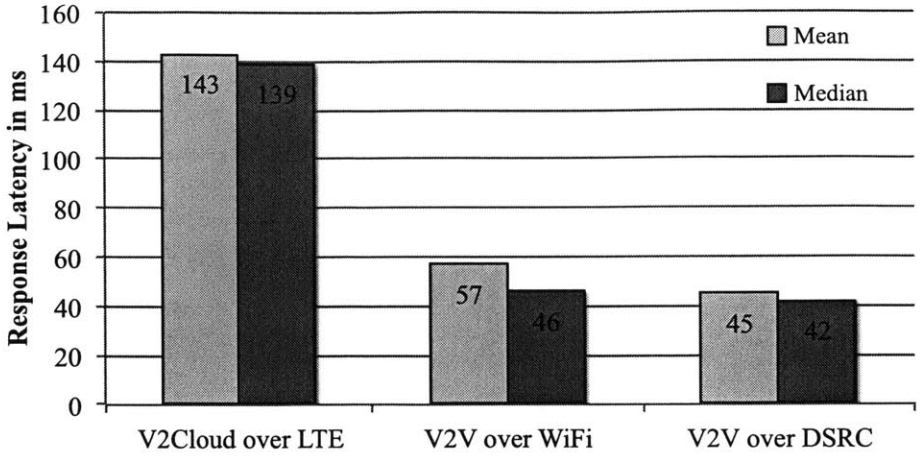


Figure 3-5: End-to-end latencies of V2Cloud, V2V-WiFi, and V2V-DSRC requests and responses.

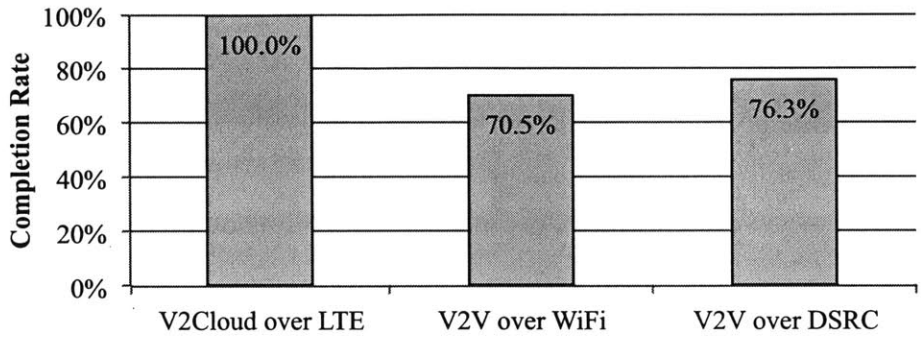


Figure 3-6: End-to-end completion rate of requests made over V2Cloud, V2V-WiFi, and V2V-DSRC.

to-head comparison; our deployment (Section 3.4) later enables a direct comparison by maintaining the same experimental settings across measurements.

V2V latencies are significantly lower than V2Cloud latencies, with up to 68.5% reduction in mean latency and 70.0% reduction in median latency.

Request completion rates are shown in Figure 3-6. V2V request completion rates are lower than V2Cloud completion rates. Because requests (the first message of two in each token exchange) do not have a retries or multiple transmissions, they have lower reception rates than responses, which are sent 3 times to prevent token loss, evident in Figure 3-6.

These findings are consistent with prior characterization studies for cellular data networks [37, 33], phone-based adhoc WiFi [26], vehicular WiFi [35, 52], and DSRC [8].

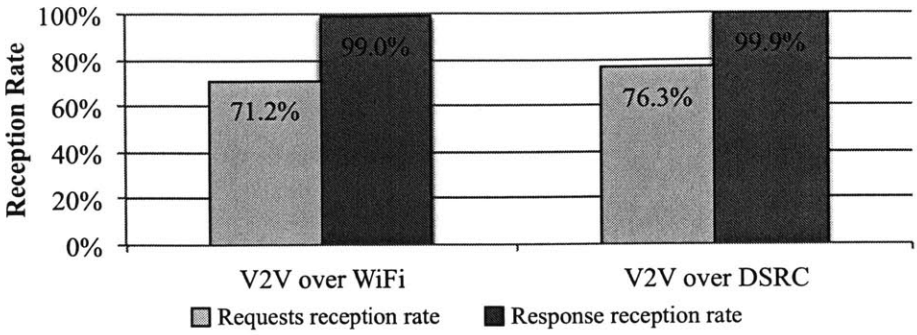


Figure 3-7: V2V-WiFi and V2V-DSRC request reception rate and response-to-request reception rate.

In summary, these experiments point to the potential benefits of leveraging V2V communications for a distributed token protocol over a conventional client-server implementation where all token requests and responses go through the cellular network to a central server, but the distributed token protocol will need to be able to accommodate the unreliability of V2V networking.

3.2.3 Distributed RoadRunner protocol

We thus designed a distributed token exchange protocol for RoadRunner that leverages V2V communications to pass tokens directly between cars where possible, falling back to a central server through the cellular network only as a backup.

A vehicle can make requests for tokens, and/or offer tokens that it is not using and does not anticipate using. The requests and offers are contained in a vehicle status report that is broadcast periodically to any neighboring vehicles in communications range. If another vehicle responds to a request with an offered token or can make use of an offered token, the vehicles exchange tokens.

This is only possible when two vehicles are within communications range of each other. As a backup, when a vehicle needs a token but cannot find an available token from cars in range, or if it has an available token but is unable to find another vehicle that demands it, it will contact the remote server via cellular to request/return a token.

There are three types of V2V packets in RoadRunner: *ANNOUNCE*, *TOKEN_REQUEST*,

and *TOKEN_SEND*. Each vehicle broadcasts an *ANNOUNCE* packet every 2 seconds, which contains the vehicle's ID, location, speed, bearing, region IDs of tokens requested by the vehicle, and region IDs of tokens currently offered by the vehicle. When a requesting vehicle receives an *ANNOUNCE* that contains an offer for a token that it needs, it sends a *TOKEN_REQUEST* packet containing the region ID that it wants. The offering vehicle removes the token from its offers list, and sends back a *TOKEN_SEND* packet containing the token and a nonce. It sends this packet three times to increase probability of reception. The requesting vehicle uses the nonce to identify and discard duplicate packets.

3.2.4 RoadRunner Walkthrough

At the beginning of a trip, a vehicle determines its route to a given destination and which regions under congestion control it must traverse. For each of these regions, the vehicle contacts the server over the cellular data connection to see if any tokens are available for those regions. If a token is available, it downloads and removes the token from the server and stores it in a collection called tokensInUse. Otherwise, it places the unfulfilled request in a retry queue. The list of unfulfilled requests is included in the *ANNOUNCE* packet which is periodically broadcast to any nearby vehicles (every 2 seconds in our deployment), and occasionally retried to the server in case any tokens are newly available on the server.

Entering a congestion-controlled region

When the vehicle enters a congestion-controlled region, one of the following scenarios occurs depending on whether it has a valid token for the region or not:

WITH a valid token. As the driver continues along the route and encounters a congestion-controlled region, it checks whether it has the corresponding token in its tokensInUse collection. If it does, then the token is marked *in-use* and the vehicle may drive through the region without any penalty or infraction logged.

Upon exiting the region, the vehicle removes the *in-use* designation from the token

and places it into a tokensOffered collection. A list of regions for which this vehicle is offering tokens is included in the periodic *ANNOUNCE* broadcast.

WITHOUT a valid token. If a corresponding token is not available for a region, RoadRunner excludes that region from its navigation calculation and will not choose routes going through that region.

If a corresponding token is not available for the region and the driver enters the region anyway, RoadRunner logs an infraction of the congestion control policy and enforces a penalty. A *PENALTY* token for that region is created and marked *in-use*. This *PENALTY* token serves to allow the driver to continue driving in that region, and possibly exit and renter it multiple times without incurring multiple penalties, expiring after a certain amount of time (10 minutes in our experimental deployment).

Upon exiting the region, the *PENALTY* reservation for this region is stored in a penaltyTokens collection so that the driver may reuse it for this region until it expires without incurring additional penalties (since the penalty for this region has already been paid).

Vehicle-to-Vehicle interactions

If vehicle A receives an *ANNOUNCE* broadcast from another vehicle B over the V2V radio (whether adhoc Wi-Fi or DSRC), vehicle A checks whether vehicle B is offering any tokens that vehicle A has unfulfilled requests for. If it does, for each of these unfulfilled requests, vehicle A sends a *TOKEN_REQUEST* message to vehicle B with the region that it wants.

Upon receiving a *TOKEN_REQUEST* message, vehicle B checks whether it has a token for the request region in its tokensOffered collection. If it does have a corresponding token, vehicle B removes the token from its tokensOffered collection and sends it back to A in a *TOKEN_SEND* message.

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. To ensure that no duplicates occur on the intended vehicle due to retransmissions, each *TOKEN_SEND* message also includes a per-vehicle nonce so that extra receptions of the same *TOKEN_SEND* message can be identified and discarded.

If multiple vehicles send a *TOKEN_REQUEST* in response to an *ANNOUNCE* message, the offering vehicle processes *TOKEN_REQUEST* messages in the order received, removing tokens from the tokensOffered collection as it goes. If it reaches a *TOKEN_REQUEST* message and does not have the requested token anymore (since it handed it over in response to an earlier *TOKEN_REQUEST* message), it ignores that *TOKEN_REQUEST* message.

ANNOUNCE packets are not rebroadcast or flooded through the system because beyond 1-hop, the latency of a V2V token exchange would exceed that of a V2Cloud token exchange, and result in a lower completion rate. Assuming a 40-millisecond latency for a single message from vehicle to vehicle (estimated from our 2-message response latency microexperiments), a 2-hop token exchange incurs 4 messages total, resulting in a latency of 160 ms vs V2Cloud's 140 ms latency.

3.2.5 RoadRunner design parameters

We describe the design parameters of RoadRunner that impact token utilization/reuse, and in turn, the effectiveness (latency and throughput) of the congestion control policy.

1. **On-demand requests versus pre-reserve requests.** RoadRunner can operate in two modes:

   Pre-reserve requests allow RoadRunner to request all the tokens it needs for a route at the beginning of a trip. Even if RoadRunner does not obtain all the tokens it needs at the beginning, it will continue attempts to fulfill any remaining requests as the vehicle continues on its route. This increases the frequency of token requests, as the vehicle keeps trying from the beginning of

the trip until it enters the region. Vehicles hold tokens for a longer period of time without actively using them, too, 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 allow RoadRunner to delay making token requests until just before it needs a token. This reduces the frequency of token requests as the vehicle does not continually attempt unfulfilled token requests from the beginning of the trip, and reduces the length of 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, imposing uncertainty.

These two modes represent a trade-off between providing the driver more certainty about the route he will take at the beginning of a trip vs. reducing the time tokens spend not in use but unavailable to other cars.

2. **Server retry timeout for unfulfilled requests.** Any unfulfilled token requests are periodically retried to the server. Longer retry periods reduce cellular data accesses, while shorter retry periods decrease the time that an available token will sit unused on the server. For the deployments, we used a server retry timeout of 2 seconds for the cloud-only variant, 10 seconds for the on-demand variant, and 30 seconds for the pre-reserve variant, determined empirically: for the cloud-only variant, any available tokens are always on the server, so that is the only place to check. For the V2V-enabled on-demand variant, any available tokens may be on other vehicles and not necessarily the server, so we don't check the server as often. For the V2V-enabled pre-reserve variant, tokens are even more likely to be available on neighboring vehicles rather than on the server because all the available tokens are grabbed from the server in advance by the vehicles, so we check the server even less often.

3. **Timeout for returning tokens.** Any tokens that the vehicle has finished

using (detected when it exits a region) and no longer needs is returned to the server after a timeout. A longer delay until returning the token allows the vehicle more time to encounter another vehicle that can receive the token over V2V communications, while a shorter delay decreases the time that a token might sit unused on the vehicle. For the deployments, we used a 10 second timeout for returning the tokens on the V2V-enabled on-demand variant, and a 60 second timeout on the V2V-enabled pre-reserve variant, determined empirically: For the on-demand variant, there are fewer oustanding unfulfilled requests in the system overall because requests are made just-in-time / on-demand when nearing a region, while for the pre-reserve variant, there are more outstanding unfulfilled requests in the system overall because each vehicle attempts to grab multiple tokens at the beginning of the trip.

3.2.6 RoadRunner metrics

Token utilization. This is the proportion of time that a token spends on a car inside a controlled region, and is directly related to region capcaity utilization. A higher token utilization implies that the number of vehicles in a controlled region is close to or at its quota, resulting in higher traffic throughput while avoiding congested conditions.

Token request fulfillment time. This is the delay from when a car initially makes a token request, until when it finally obtains a token, perhaps after several retries to the server or after a V2V interaction with a nearby vehicle. Longer token fulfillment times may be necessary to maintain a quota and reduce congestion on controlled regions, but are often undesirable, as they can result in lowered throughput through the transportation system as cars are throttled waiting for tokens. They can also result in more retries to the server, thus adding load on the cellular connection.

Ultimately, token reuse is desirable, and underpins the benefits of RoadRunner's distributed road reservation protocol. When tokens can be circulated and reused among cars directly, cellular accesses for requesting tokens are offset, and token fulfillment times can be lowered when cars can obtain a token from another car rather

than waiting on a periodic server retry. To enable token reuse, however, cars that are done using their tokens must hold onto the tokens for a period of time to make them available to other cars over V2V. This results in fewer free tokens being available on the server.

3.3 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 (adhoc Wi-Fi or DSRC), running a busy-wait loop to receive packets from the network interface associated with the V2V radio.

3.3.1 V2Cloud Communications

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).

3.3.2 V2V Communications

For V2V communications, the app can leverage either 802.11n adhoc Wi-Fi or 802.11p DSRC. We implemented support for both interfaces in our Android application.

V2V over 802.11n Wi-Fi

To use 802.11n adhoc Wi-Fi, we run Android 2.3 Gingerbread on the Galaxy Note smartphones to use adhoc Wi-Fi because only the Gingerbread drivers for the Broadcom BCM4330 chipset in the phone support wireless extensions (WEXT). We use a cross-compiled iwconfig binary to configure the cards in adhoc mode at the lowest bitrate supported (1Mbps) with power management turned off. Each smartphone is configured with a unique IP address, and all V2V communications is done over UDP broadcast.

V2V over 802.11p DSRC

To use 802.11p DSRC, we connect the Android smartphone to a Cohda Wireless MK2 WAVE-DSRC Radio [18]. The MK2 provides a USB 2.0 host interace through the use of a mini USB On-The-Go (OTG) adapter, to which we connect the Android smartphone with a microUSB cable. We enable USB tethering on the Android smartphone, which presents the Android smartphone as a generic USB ethernet adapter (USB CDC Ethernet) to the MK2 host, allowing for ethernet frames to be transmitted between the two devices.

The MK2 runs FwdWsm, a software bridge application that receives UDP packets on the USB 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. This allows Android applications to communicate over the 802.11p radio simply by sending and receiving UDP packets on the USB Ethernet interface.

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.

3.3.3 Implementation discussion

Several practical considerations of implementing a distributed congestion control system are discussed below.

Electronic tolling/road-pricing. A time-based road pricing scheme, such as that used in Singapore's ERP [74], 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.

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 [74] system, or it can be electronically managed via online accounting systems like PayPal.

From another perspective, RoadRunner already operates as a hybrid road-pricing and road-space rationing scheme, as drivers could intentionally drive onto roads without a token and choose to pay the penalty, which could change according to time-of-day. These penalty charges could be updated over time as well.

Security. In existing electronic congestion control schemes, the security of the system is dependent on the presence of a trusted in-car unit in each vehicle, and tampering with the unit is prohibited by regulations policy [74]. The RoadRunner app runs on a user's smartphone, an untrusted host, and we want to prevent malicious users from forging and/or duplicating tokens or spoofing their location to misreport presence in a region, which can be addressed by prior work on the trustworthiness of a reported location [46]. Claessens et al [17] remark that the most difficult problem in doing secure electronic transactions on mobile devices is that of the untrusted host; in this case, we want to protect the transportation authority from a mobile device's user, known as undercover-agent trust [61]. RoadRunner can use digital signatures to verify that tokens are legitimate, requiring that the module verifying signatures to

be trusted, e.g. the implementation by Schmidt et al [72] of the Trusted Computing Group's Mobile Trusted Module specification, or by Winter et al [85] on embedded linux-based ARM trustzone platforms.

Privacy. Lenders et al [46] also describe a decentralized certificate authority for their secure localization service, which can preserve the privacy of users generating location-tagged information (such as the *ANNOUNCE* messages).

GPS localization in urban cities. GPS localization faces problems in urban environments with tall buildings (called urban canyons). To overcome this challenge, Cui et al [19] successfully demonstrated a vehicle path-constrainment method, and Vicek et al [81] combined GPS with a gyro and vehicular odometer, techniques which are orthogonal to RoadRunner.

3.4 Deployment of RoadRunner Prototype

We implemented a mobile application to evaluate three variations of RoadRunner: a Cloud-only variant that communicates solely with a remote server, a Wi-Fi-enabled variant that communicates with a remote server and with other vehicles over adhoc Wi-Fi, and a DSRC-enabled variant that communicates with a remote server and with other vehicles over DSRC / 802.11p. 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.

For the V2V-enabled variations, 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 begin checking with the cloud at the beginning of a trip or on-demand.

Our deployment took place in eastern Cambridge, Massachusetts, USA (Figure 3-10). 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 3-9). This resulted in a fine-grained congestion control scenario of 4 controlled regions with a total distance of 900 meters. For comparison, Singapore's Orchard Road ERP zone is 1 controlled region of distance 2200 meters, an order of magnitude larger.

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 will provide 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 3-8. 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 permutations of RoadRunner's modes.

3.4.1 Request fulfillment offload

Figure 3-11 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

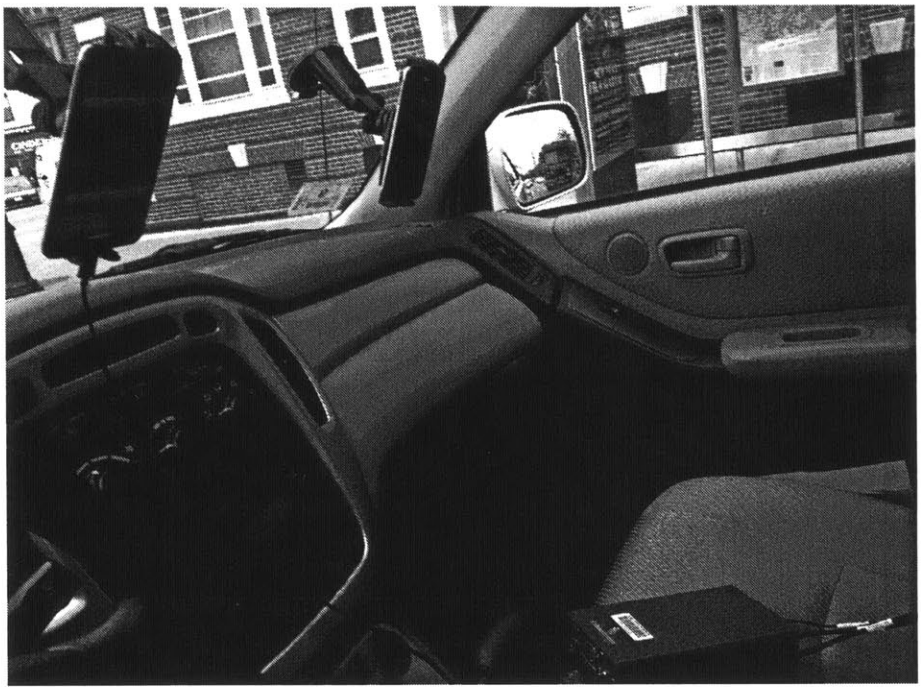


Figure 3-8: Picture of deployment setup in each vehicle.

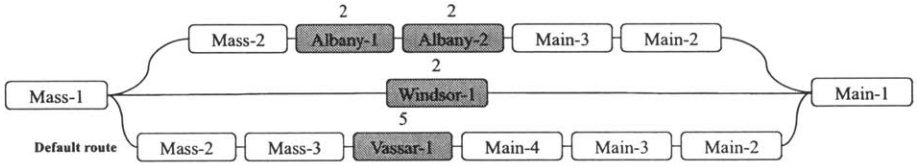


Figure 3-9: Routes in our deployment, with controlled regions shaded and their capacity shown above each block.

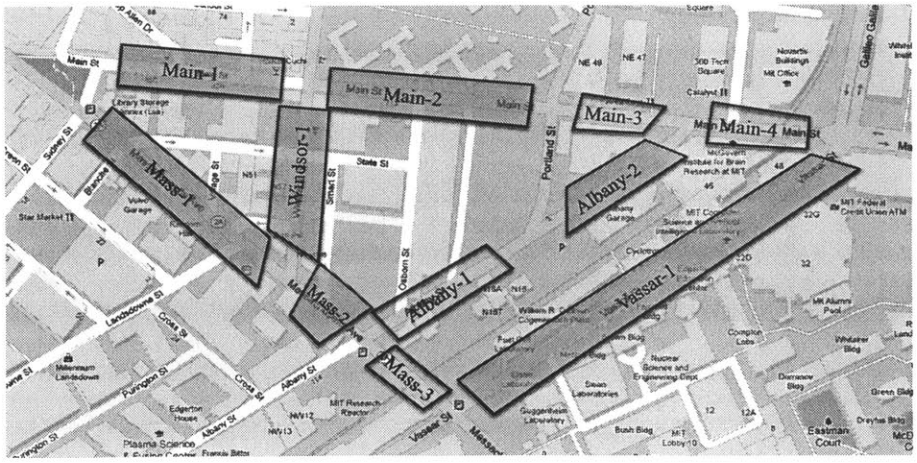


Figure 3-10: Map of deployment area and regions.

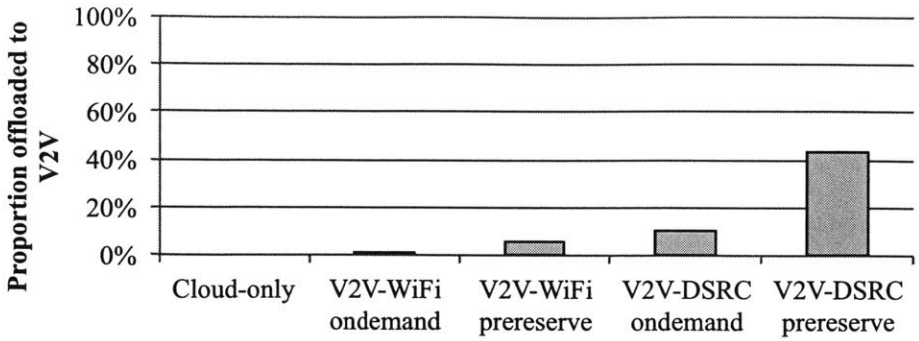


Figure 3-11: Proportion of all fulfilled requests over V2V.

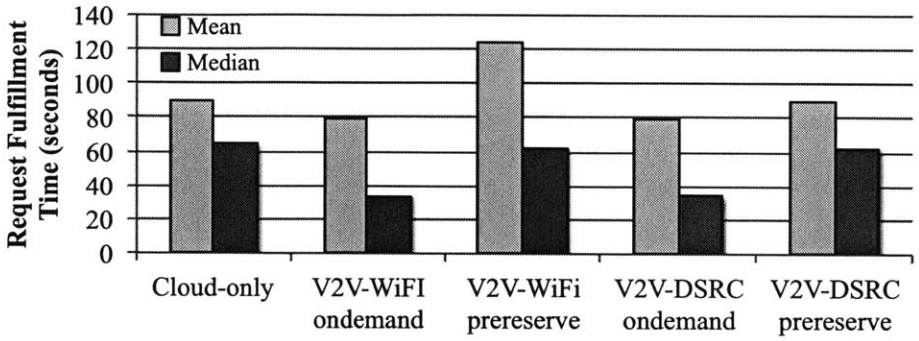


Figure 3-12: Time to token request fulfillment.

than just before the region, giving vehicles more time to encounter a token offered over V2V.

3.4.2 Request fulfillment time

All token requests are timestamped from when the request is created to when the request is fulfilled, whether by the centralized server (V2C) or another vehicle over V2V communications. We graph the fulfillment times for the variants of RoadRunner in Figure 3-12.

Average request fulfillment times for DSRC and WiFi variants of RoadRunner are similar to the Cloud-only baseline, with the exception of the V2V-WiFi pre-reserve variant. This variant obtained only 5 tokens over V2V out of 73 total tokens obtained (a 6.8% ratio) due to the limited range of WiFi, while V2V-DSRC pre-reserve obtained 37 tokens over V2V out of 86 total, a much higher 43% ratio. DSRC's improved range allowed it to more often short-circuit the wait for a token to appear on the cloud,

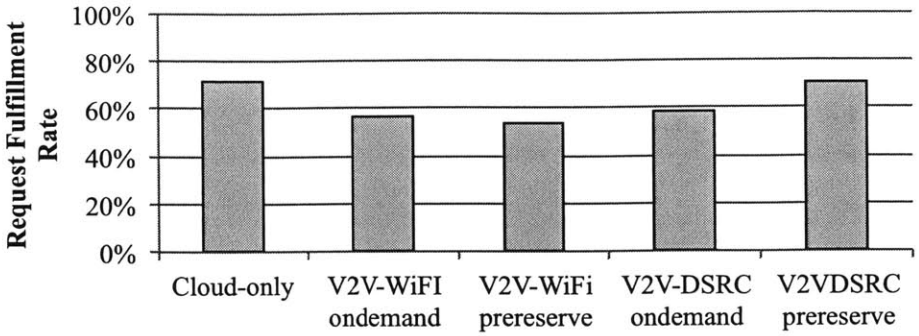


Figure 3-13: Request fulfillment rate

reducing its fulfillment time. Thus, the distributed token reservation protocol is able to match the performance of the Cloud-only baseline in fulfillment time with good V2V communications. 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.

3.4.3 Request fulfillment rate

The fulfillment rate is the proportion of token requests that eventually end in the successful acquisition of a token. In a congestion control system, a fulfillment rate of 100% is undesirable since that would imply allowing every vehicle into a controlled region, which implies no congestion control. The fulfillment rate is useful for understanding the effects of the more unreliable vehicle-to-vehicle communications and token exchange protocol vs. the usually available cellular and cloud server (which was available 100% of the time in our microexperiments, and 91% of the time in our deployment).

Since the total number of vehicles in the system and the total number of tokens per region is held constant throughout our experiments, we should expect similar fulfillment rates from the V2V-enabled variations of RoadRunner. Any significant deviations would imply that the RoadRunner system itself is 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 3-13. DSRC RoadRunner show similar fulfillment rates to the Cloud-Only baseline, imply-

ing 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).

3.4.4 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 chooses to take a different route or is unable to turn onto the new route in time, RoadRunner makes the tokens available to other vehicles or puts them back on 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 3-14. 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 base-

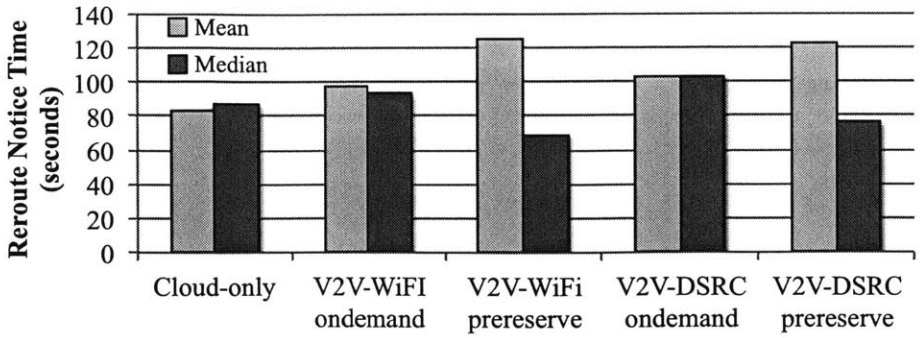


Figure 3-14: Reroute advance notice time available to driver when a more preferable route becomes availabe.

line in reroute time provided to the driver. The pre-reserve V2V variants had a bimodal distribution: when vehicles are able to prereserve 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.

3.4.5 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 occuring 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 3-15, 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 char-

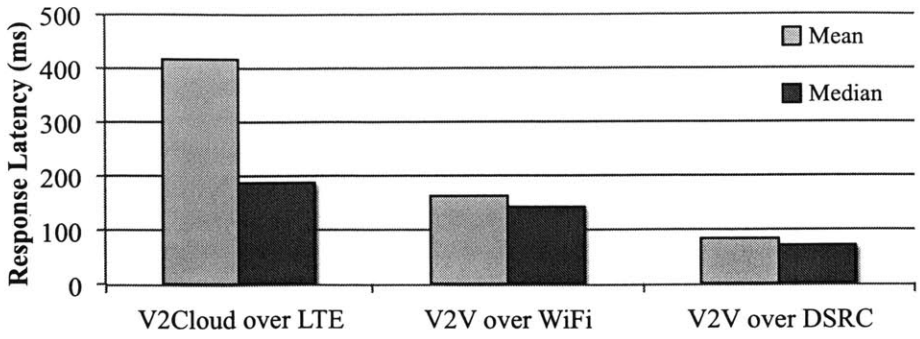


Figure 3-15: System end-to-end response latency to requests, whether requests are fulfilled or not.

acterization studies [33, 42, 71].

DSRC latencies are not as low as the 100 microseconds delay requirement for safety applications or previously measured DSRC latencies [89], 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 [51] [45] [79] that rely solely on cellular.

3.4.6 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 3-16 shows that all the V2V variants of RoadRunner are able to reduce the number of cloud accesses per token significantly compared to the Cloud-only variant with reductions ranging from 66.3% to 84.3%.

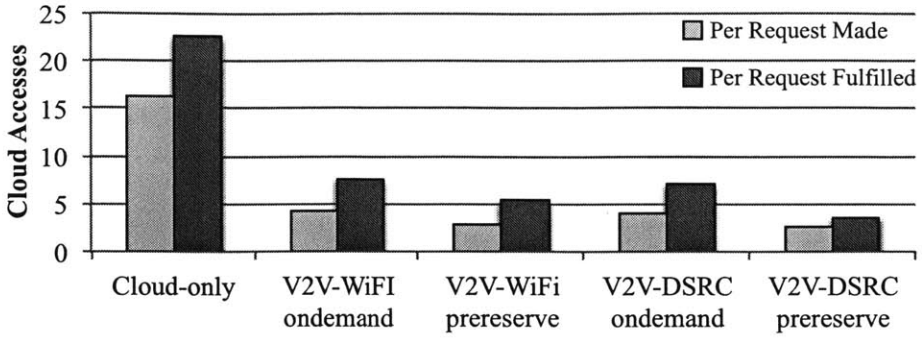


Figure 3-16: Cellular data accesses to the cloud server divided by number of requests made, and by number of requests fulfilled.

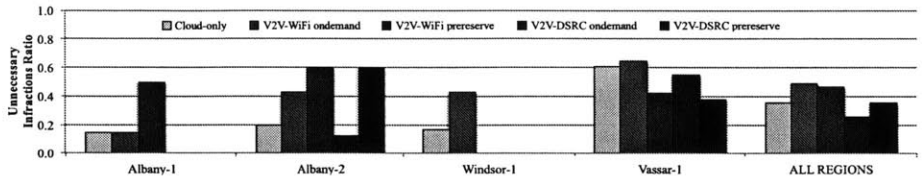


Figure 3-17: Ratio of unnecessarily penalized entries to total entries. Certain variants experienced no unnecessary infractions in regions.

3.4.7 Overall congestion control effectiveness

We evaluated the ability of RoadRunner to ensure that there are no more vehicles with valid tokens in each region than was originally allotted, and that any vehicles without a valid token incur a penalty reservation.

A system that is perfectly efficient would enforce an upper bound on the number of vehicles in a region by allowing all vehicles to enter without penalty up to the upper bound, and once the maximum capacity of the region is reached, would penalize all vehicles entering above the upper bound. Vehicles may be unecessarily penalized for entering a region even when the region has not yet reached capacity, due to another vehicle posessing an unused or not-yet used token for that region, making it unavailable to a vehicle that could have used the token earlier. We calculate the unnecessary infraction ratio as the number of unnecessary infractions divided by the total number of vehicle entries into a region for each controlled region, and for all controlled regions, in Figure 3-17.

V2V-WiFi consistently incurs more unnecessary infractions than both Cloud-only and V2V-DSRC, because free tokens are effectively tied up on the cars as V2V offers

that never get heard due to the limited WiFi range, making them unavailable to other cars checking the server. V2V-DSRC has sufficient V2V communications range and does not run into this problem, as reflected in Figure 3-17.

We see that even in the Cloud-only baseline, many unnecessary infractions occur. This is due to the very low number of tokens available for each region and the total system (only 2-5 per region), which is a consequence of our limited deployment size of 10 vehicles. In our Cloud-only simulation (Section 3.5), we see that infractions begin to be enforced when the region has not quite reached capacity; this effect happens here as well, except that we have few vehicles in the first place. Instead, we rely on our large-scale simulation to demonstrate the enforcement effectiveness of RoadRunner's distributed road reservation protocol.

In summary, these results demonstrate the benefits of leveraging V2V communications for token exchange interactions among vehicles, rather than a conventional client-server implementation where all token requests and responses go throu-gh the cellular network to a centralized server. We achieved reductions in cloud accesses incurred per request by up to 84% and reductions in response latencies by 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. In the following simulation studies, we show that at large scale, RoadRunner incurs much fewer unnecessary infractions in all variants than in our limited deployment size.

3.5 Large-Scale Simulation of RoadRunner

We also evaluated the operation of RoadRunner at scale by using loop detector counts from Singapore to simulate system operation over a 24 hour period beginning Sunday, August 1, 2010, resulting in 165,272 boundary crossings across the entire Orchard Road Region. Vehicle counts are available for nine intersections on Orchard Road (Figure 3-18), a road region that experiences heavy traffic volume and is currently under electronic road pricing congestion control as a single controlled region. We sim-

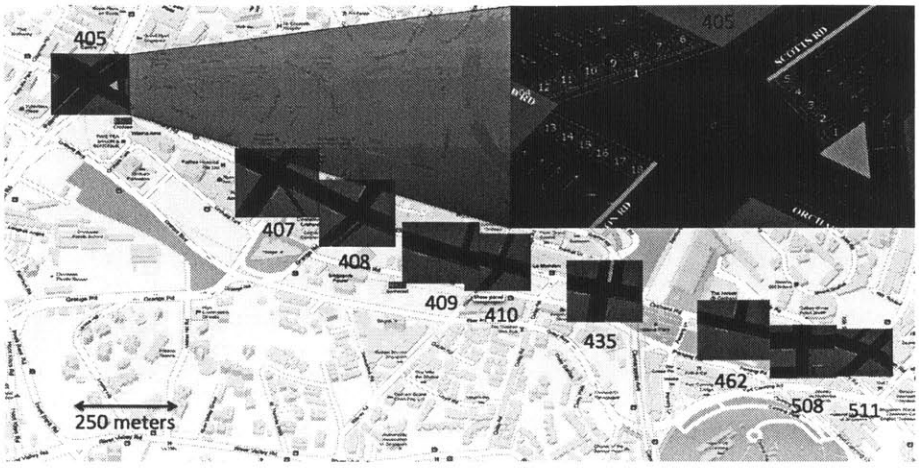


Figure 3-18: Map of Orchard Road region and intersections used for simulation.

ulate Orchard Road as a single controlled region in RoadRunner as well, to motivate the viability of RoadRunner as a drop-in replacement congestion control scheme to this real-world congestion control deployment.

For each intersection, we manually annotated which detectors counted vehicles turning onto (entering) the region, which detectors counted vehicles turning out of the region (exiting), and which detectors counted vehicles continuing to travel inside the region.

We simulated the RoadRunner on-demand protocol with a congestion control policy providing 3000 road reservation tokens on Orchard Road. We did not simulate RoadRunner pre-reserve as the loop count data does not provide locations of trip origins. Since the loop count data is prerecorded, RoadRunner cannot reroute vehicles that have not obtained a token, but it does detect them as infractions once they enter the region without a token.

3.5.1 Vehicle movement model and communications model

The raw loop count data provides the number of vehicles crossing through the intersection of each lane every 5 minutes through the use of loop detectors embedded under each lane. To model the movement of vehicles within the region, we first interpolate loop count data to a higher temporal resolution (necessary to simulate the

V2V communications) by modeling vehicle arrivals at each loop count detector as a Poisson process and distributing these arrivals uniformly across the 5 minute time interval. We then bin them into 30 second intervals, thus producing interpolated loop count data with a temporal resolution of 30 seconds rather than 5 minutes.

We model 9 subregions, 1 for each of the intersections. With each subregion, we model the movement of vehicles at a speed of 30 km/h when moving, and assume the vehicle is in a stop-and-go traffic pattern moving half the time, for an average speed of 15 km/h with entering vehicles traveling away from the intersection into the region, and vehicles exiting the region traveling away from the region boundary. In each 30 second timestep, these vehicles can travel up to 125 meters, providing us with a bounded estimate of their distance from the intersection. Any entering vehicle is thus at most 125 meters away from any exiting vehicle and vice-versa, and any vehicle is at most 250 meters away from any other vehicle.

We assume a V2V communications range of 125 meters with a 100% message reception rate, based on the average DSRC token exchange distance of 175.7 meters in our real-world deployment (Section 3.4). Thus, in each timestep, all vehicles entering and exiting a region are able to communicate with each other and exchange any tokens.

We also simulate a cloud server that exchanges tokens with the vehicles over a V2Cloud cellular connection, assuming this connection is available 100% of the time. The cellular data connection was available 100% of the time in our microexperiments, and intermittently available 91% of the time in our deployment (100% available after retries).

Our simulated V2V and V2C communications are instantaneous, as the purpose of these simulations is to evaluate the viability of RoadRunner's distributed token reservation protocol at scale with realistic vehicle traffic patterns and densities.

3.5.2 Simulation initialization and iteration

We initialize the simulation by placing 1500 vehicles within the region when the simulation begins at 12:00am. These vehicles all start with a valid token. (Over the

course of the simulation, this yields a maximum of 3606 vehicles in the entire Orchard Road region at 7:26pm and a minimum of 117 vehicles at 9:54am.)

At each timestep we simulate the RoadRunner distributed token reservation protocol at each intersection as follows:

1. Entry/exit calculation. We calculate the number of cars entering and exiting the subregion/intersection by summing the loop counts for the relevant lanes in which cars enter and exit. For lanes with more than one possible outcome (e.g. a lane that allows turning left to exit the subregion or continuing forward and remaining in), the car randomly chooses a path with equal probability.

2. V2V token exchange. Vehicles exiting the subregion in the opposite direction send their tokens to vehicles entering the subregion over V2V communications.

3. V2C token exchange. Any entering vehicles that were unable to obtain a token over V2V request a token from a cloud server, which returns a token if it has any. Any exiting vehicles that still have tokens send them back to the cloud-server. Note that even with perfect V2V message reception at each subregion, RoadRunner still incurs cloud accesses: if there are no tokens being offered by exiting vehicles or extra tokens after the token exchange simulation, requests for getting and returning tokens must go to the cloud.

At the end of each timestep, we calculate statistics for each of the 9 simulated subregions, including boundary crossings, entering and exiting vehicle counts, infractions, tokens exchanged over V2V, and tokens exchanged over V2C. We also compute statistics for the entire Orchard Road region composed of these 9 subregions.

3.5.3 Simulation results

Quota enforcement and region utilization

We show the region utilization over time of Orchard Road in Figure 3-19 for a Cloud-only baseline, and in Figure 3-20 for a DSRC-based V2V range of 125m. Note that

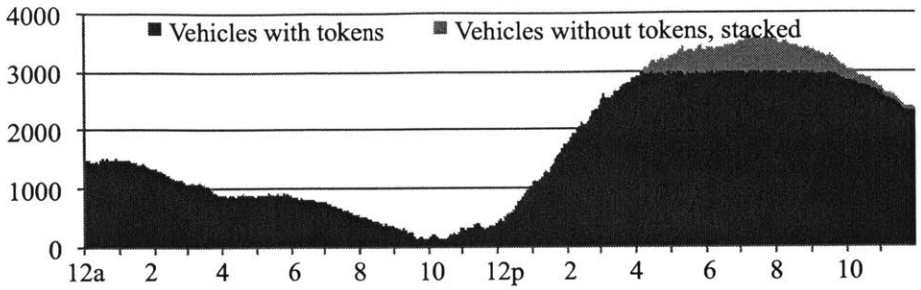


Figure 3-19: Orchard Road vehicle count over course of Cloud-only simulation.

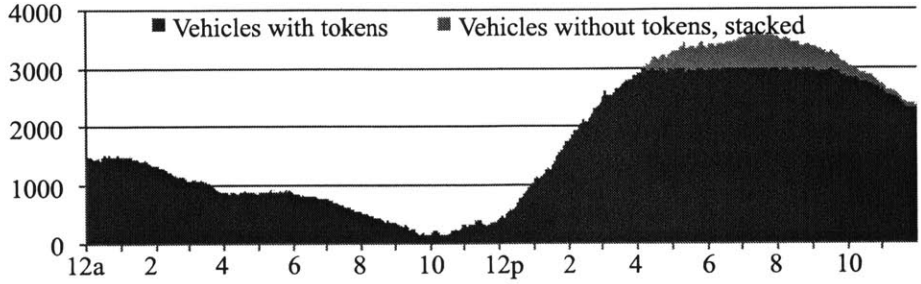


Figure 3-20: Orchard Road vehicle count over course of DSRC V2V simulation.

RoadRunner successfully enforces the upper limit on the road at all times: the number of vehicles with tokens in the region never rises above 3000.

Out of 83,075 vehicle entries, 3.03% (2517) of all vehicle entries have unnecessary penalties in the Cloud-only baseline. These occurs when the cloud runs out of tokens and an exiting vehicle returns a token to the Cloud too late for another entering vehicle that has already requested a token from the Cloud. When V2V is enabled, this drops to 2.96% (2461) since some vehicles that couldn't get a token from the cloud are now able to obtain a token over V2V instead. These unnecessary infractions are now due to the limited V2V range which does not allow for token exchanges across multiple intersections. These results show that at real-world vehicle densities in a controlled region, the use of our distributed token reservation protocol is able to reduce unnecessary penalties.

Cellular data access reduction

In the Cloud-only baseline, 160,249 V2Cloud interactions or cellular data accesses occured, whether to GET or PUT tokens. With V2V enabled, 23,735 V2V interactions

occured, offloading and **reducing cellular data accesses by 29.6%** to 112,836. The cellular data access reduction is lower than in the evaluation because we do not simulate the periodic server timeout, as the loop count data does not provide traces for cars extending beyond the region, which are necessary to simulate the periodic server checks as the car approaches Orchard Road.

3.6 Related Work

3.6.1 Infrastructure-less congestion control

RoadRunner is related to several previously demonstrated systems in infrastructure-less congestion control, but differs from all of them by leveraging direct V2V communications instead of relying solely on a cellular data connection. Lu et al [51] built a GPS-based tolling system using iPaq PocketPCs and portable GPS receivers, with tolling information reported through a GPRS connection to a backend server. Lee et al [45] built an electronic tolling system using GPS and 3G for localization and communication, which sends tolling information over a 3G cellular connection. Srinivasan et al [79] presented 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.

3.6.2 Vehicular networks

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 routing of messages, vehicular positioning, and security. Leontiadis et al [48] present a geographic routing protocol for vehicular networks and simulate using vehicle traces. Wu et al's MDDV [87] 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 [14] routes message between peers without knowing the state

of a partitioned disruption-tolerant network or the meeting locations. Wisitpongphan et al [86] 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 [12] examine the suitability of data fusion techqniues to provide robust localization for vehicular networks, which could help improve our controlled region granularity. Parno et al [60], Raya et al [66], and Lin et al [49] contribute protocols, discussion, and designs on securing vehicular networks, critical to ensuring malicious users do not defraud or disable RoadRunner.

Chapter 4

Conclusion

Our two system prototypes show that mobile applications running on widely-available smartphones can achieve significant improvements in responsiveness and cellular bandwidth consumption by leveraging mobile sensing, local computation, and device-to-device communications.

With DIPLOMA (Chapter 2), we demonstrated that shared memory, a programming paradigm that is widely adopted for parallel programming, can be realized on mobile devices. As mobile applications for public services such as transportation become increasingly pervasive, we envision the opportunity to piggyback systems software such as DIPLOMA onto large numbers of mobile devices, realizing a powerful mobile computing platform that can offload communications from cellular networks and computation from servers. DIPLOMA takes a step towards this vision, investigating shared memory as an alternative programming model to client-server, paving the way for ubiquitous distributed mobile computing.

With RoadRunner (Chapter 3), we demonstrated that a congestion control system can be realized with a distributed, infrastructure-less token reservation protocol that combines ubiquitous smartphones with vehicle-to-vehicle communications. Our microexperiments and deployments showed sufficient enforcement accuracy, faster response times, and effective offload of cellular accesses when compared to a traditional client-server implementation. Our simulation results show that at realistic density, such a system can effectively enforce congestion control.

Infrastructure-less ITS based on smartphones can enable very low cost, truly ubiquitous intelligent transportation services. With device-to-device networking improving rapidly and gaining widespread implementation in modern mobile devices through emerging standards such as WiFi Direct [82] and LTE Direct [63], we hope to see more mobile distributed systems and applications.

Bibliography

[1] J. C. Abela and E. M. Aguilar. Panoramio. `http://www.panoramio.com/`.

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

[3] N. Agarwal, L.-S. Peh, and N. K. Jha. In-network snoop ordering (INSO): Snoopy coherence on unordered interconnects. In *Proc. HPCA*, 2009.

[4] M. Almenoar. Call for new ERP proposals. *Straight Times*, June 2010.

[5] Alpha Architecture Committee and Richard L. Sites. Alpha Architecture Reference Manual, 1992.

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

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

[8] F. Bai, D. D. Stancil, and H. Krishnan. Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers. In *Proc. ACM Mobicom*, 2010.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proc. ACM IMC*, Nov. 2009.

[10] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *SIGOPS Oper. Syst. Rev.*, 17(5):3, 1983.

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

[12] 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.

[13] M. Brown et al. The virtual node layer: A programming abstraction for wireless sensor networks. In *Proc. Int. Wkshp. Wireless Sensor Network Architecture*, 2007.

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

[15] D. Chen et al. Interweave: A middleware system for distributed shared state. In *5th International Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers*, 2000.

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

[17] J. Claessens, B. Preneel, and J. Vandewalle. (how) can mobile agents do secure electronic trans. on untrusted hosts? a survey of the security issues and the current solutions. *ACM Trans. on Internet Technology*, 3(1):28–48, 2003.

[18] Cohda Wireless. MK2 WAVE-DSRC Radio. `http://cohdawireless.com/product/mk2.html`.

[19] Y. Cui and S. Ge. Autonomous vehicle positioning with GPS in urban canyon environments. *IEEE Trans. on Robotics and Automation*, 19(1):15–25, 2003.

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

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

[22] S. Dolev et al. Geoquorums: Implementing atomic memory in mobile ad hoc networks. *Distrib. Comput.*, 18(2):125–155, 2005.

[23] J. Dunn. Cars Get Connected: The Best Car Tech Trends Today. `http://www.technologyguide.com/default.asp?newsID=5255`, Dec. 2012.

[24] K. Fall and K. Varadhan. The network simulator (ns-2). 2007.

[25] J. FINN. The rebirth of toll: Etc makes its american comeback. *Tolltrans. Traffic Tech. Intl. Supplement*, 2004.

[26] J. Gao, A. Sivaraman, N. Agarwal, H. Li, and L. Peh. Diploma: Consistent and coherent shared memory over mobile phones. In *Proc. IEEE ICCD*, 2012.

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

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

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

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

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

[32] 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.

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

[34] IBM Corporation. IBM System/370 Principles of Operation,IBM, May 1983. Publication Number GA22-7000-9, File Number S370-01.

[35] M. Jerbi, S. Senouci, and M. Al Haj. Extensive experimental characterization of communications in vehicular ad hoc networks within different environments. In *Proc. IEEE VTC*, 2007.

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

[37] J. Karlsson and M. Riback. Initial field performance measurements of LTE. *Ericsson Review*, 3:22–28, 2008.

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

[39] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Trans. Comput. Syst.*, 10, Feb. 1992.

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

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

[42] E. Koukoumidis et al. Pocket cloudlets. *ACM SIGARCH Computer Architecture News*, 39(1):171–184, 2011.

[43] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Comput.*, 28:690–691, Sept. 1979.

[44] A. Lee. Verizon Killing Unlimited Data Plans Soon. `http://www.huffingtonpost.com/2011/06/21/verizon-unlimited-data-plan_n_881091.html`.

[45] 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.

[46] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *Proc. ACM HotMobile*, 2008.

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

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

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

[50] R. Lindsey. Do economists reach a conclusion? *Econ Journal Watch*, 3(2):292–379, 2006.

[51] S. Lu, T. He, and Z. Gao. Electronic toll collection system based on global positioning system technology. In *Challenges in Environmental Science and Computer Engineering (CESCE), 2010 Intl. Conf. on*, volume 2, Mar. 2010.

[52] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based connectivity from moving vehicles. In *Proc. ACM SIGCOMM IMC*, 2007.

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

[54] Monsoon Solutions. Monsoon Power Monitor. `http://msoon.com/LabEquipment/PowerMonitor/`.

[55] 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.

[56] Nielsen Co. Netizens in singapore spend more than a day a week on internet, 2012.

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

[58] NVIDIA Corporation. Variable SMP A Multi-Core CPU Architecture for Low Power and High Performance. `http://www.nvidia.com/content/PDF/tegra_white_papers/Variable-SMP-A-Multi-Core-CPU-Architecture-for-Low-Power-and-High-Performance.pdf`.

[59] R. O'Toole. A Free Parking Space Grows in Manhattan. `http://ti.org/antiplanner/?p=3565`, 2010.

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

[61] A. Pfitzmann et al. Trusting mobile user devices and security modules. *Computer*, 30(2):61–68, 1997.

[62] 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.

[63] Qualcomm Corporation. LTE Direct. 2010.

[64] Qualcomm Inc. Snapdragon Processors. `http://www.qualcomm.com/chipsets/snapdragon`.

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

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

[67] Rysavy Research, LLC. Mobile Broadband Capacity Constraints And the Need for Optimization. `http://www.rysavy.com/articles/2010_02_Rysavy_Mobile_Broadband_Capacity_Constraints.pdf`, 2010.

[68] Samsung Corporation. Samsung Galaxy Note S3. `http://www.samsung.com/global/business/data/Exynos420QUAD.pdf`.

[69] G. Santos and G. Fraser. Road pricing: lessons from london. *Economic Policy*, 21(46):263–310, 2006.

[70] 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.

[71] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[72] A. Schmidt, N. Kuntze, and M. Kasper. On the deployment of mobile trusted modules. In *Wireless Comm. and Networking Conf.*, 2008.

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

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

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

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

[77] SPARC International. The SPARC Architecture Manual, Prentice Hall, 1994. SPARC International,Version 9.

[78] D. Sperling and D. Gordon. *Two billion cars: driving toward sustainability.* Oxford University Press, USA, 2009.

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

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

[81] C. Vicek, P. McLain, and M. Murphy. GPS/dead reckoning for vehicle tracking in the urban canyon environment. In *IEEE Vehicle Navigation and Information Systems Conf.*, 1993.

[82] Wi-Fi Alliance. Wi-fi peer-to-peer (P2P) technical 7 specification. 2010.

[83] Applications of road space rationing. `http://en.wikipedia.org/wiki/Road_space_rationing`.

[84] List of Electronic Toll Collections Systems. `http://en.wikipedia.org/wiki/List_of_electronic_toll_collection_systems`.

[85] J. Winter. Trusted computing building blocks for embedded linux-based arm trustzone platforms. In *Proc. ACM STC*, 2008.

[86] 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 Comm.*, 25(8):1538–1556, 2007.

[87] 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.

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

[89] 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.

[90] 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.