On Demand Surveillance Service in Vehicular Cloud

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Jui-Ting Weng

2013
© Copyright by
Jui-Ting Weng
2013
ABSTRACT OF THE DISSERTATION

On Demand Surveillance Service in Vehicular Cloud

by

Jui-Ting Weng

Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2013
Professor Mario Gerla, Chair

Environment monitoring and surveillance using video cameras and sensors have become increasingly important functions in urban centers. Specifically, vehicles are ideally suited for surveillance complementing fixed video cameras and sensors installed in the infrastructure. As on-board video cameras become more popular in public transportation and individually owned vehicles, these devices provide an excellent source for surveillance data. When these vehicles collaborate between each other and provide on demand video services, the vehicular ad hoc forms a service, which provides picture and video service to any client needing the data. Comparing with currently existing cloud network, the vehicle is no longer a cloud service consumer; instead, it becomes the hardware provider to offer location base video and picture services. The service utilizes hardware APIs such as GPS, event recording cameras, and network connections. As a result, the surveillance application exploits the mobility of the service vehicles to extend surveillance coverage beyond the reach of static devices.
The dissertation of Jui-Ting Weng is approved.

Danijela Cabric

Milos Ercegovac

Mani Srivastava

Mario Gerla, Committee Chair

University of California, Los Angeles

2013
To my parents
# Table of Contents

1 **Introduction** .................................................. 1

1.1 Toward Vehicular Service Cloud ............................... 1

1.2 Cloud computing to Vehicular Cloud Service ................. 2

1.3 Parallel experiment platform with Vehicular Cloud .......... 3

1.4 Surveillance Service with Vehicular Cloud ................... 4

2 **PepNet: Running Consistent, Parallel Experiments in Vehicular Environment** ............................. 7

2.1 Overview of Parallel experiments ............................... 7

2.2 Related Work .................................................... 10

2.3 PepNet Overview .................................................. 11

2.3.1 Hardware Platform ............................................. 12

2.3.2 Software Platform ............................................ 13

2.4 PepNet Overhead Evaluation ..................................... 14

2.4.1 Virtualization Overhead ...................................... 16

2.4.2 Sharing Overhead .............................................. 19

2.5 Field Experiments ............................................... 22

2.5.1 Experimental Setup ........................................... 22

2.5.2 Experiment Results .......................................... 27

2.6 Conclusion and Future Work .................................... 35

3 **Pics-On-Wheel: Surveillance Service in the Vehicular Cloud** ........................................... 38

3.1 Overview ......................................................... 38
List of Figures

2.1 Node setup on a vehicle .............................................. 12
2.2 Software Platform ................................................... 13
2.3 Virtualization Overhead experiment scenarios. .................... 15
2.4 Overhead introduced by Xen virtualization: packet delivery ratio while performing the VM to VM experiment with different loads of CBR traffic. ......................................................... 19
2.5 Overhead introduced by Xen virtualization .......................... 21
2.6 Xen fairness evaluation: packet delivery ratio for different Xen guest pair while running 5 VMs in parallel ......................... 23
2.7 Bird’s-eye view of topology setting for all experiment rounds ... 27
2.8 Network connectivity: optimal hop count from mobile sender to one fixed node over time for experiment round 1, 2 ............... 29
2.9 Network connectivity: optimal hop count from mobile sender to another mobile receiver over time for experiment round 7, 8 ... 30
2.10 Non-parallel experiments: rounds 1 through 4 .................... 31
2.11 Single mobile node parallel experiments: rounds 5 and 6 ...... 33
2.12 Two mobile node parallel experiments: rounds 7 and 8 ....... 34
2.13 False Evaluation: an incorrect result when two protocols running in two sequential experiments are compared (OLSR Round 7/ AODV Round 8). ................................................................. 35
3.1 Mobile Cloud System .................................................. 39
3.2 Zone of Service ......................................................... 42
3.3 location of selected targeted location ............................... 48
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Service availability comparison between per minute location update(with cabspotting location update) and full path knowledge</td>
<td>49</td>
</tr>
<tr>
<td>3.5</td>
<td>Cumulative distribution between a Service Request arrives Cloud Service Provider and the Service is completed</td>
<td>50</td>
</tr>
<tr>
<td>3.6</td>
<td>The average number of picture collected with different number of vehicles for a targeted 30-second window</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>Success rate to finish the service within 5 minutes</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>Service Location</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Active Taxi</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Pics-on-Wheel service architecture</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Pics on wheel hardware setup and photo example</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>Pics-on-Wheel Latency Breakdown</td>
<td>63</td>
</tr>
<tr>
<td>4.6</td>
<td>Location of traffic samples</td>
<td>65</td>
</tr>
<tr>
<td>4.7</td>
<td>Traffic Volume for undetoured traffic</td>
<td>66</td>
</tr>
<tr>
<td>4.8</td>
<td>Service delay provided by POW service in 5 different locations</td>
<td>66</td>
</tr>
<tr>
<td>4.9</td>
<td>Traffic Volume for taxi willing to detour for a target 100 meter away</td>
<td>67</td>
</tr>
<tr>
<td>4.10</td>
<td>On demand service latency with maximum 100m detour</td>
<td>68</td>
</tr>
<tr>
<td>4.11</td>
<td>On demand service latency with maximum 200m detour</td>
<td>69</td>
</tr>
<tr>
<td>4.12</td>
<td>Actual service delay and approximation with Poisson distribution</td>
<td>70</td>
</tr>
<tr>
<td>4.13</td>
<td>Actual service delay and approximation with rate adjusted Poisson</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>distribution on location A and E</td>
<td>71</td>
</tr>
<tr>
<td>4.14</td>
<td>Actual service delay and approximation with rate adjusted Poisson</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>distribution on location B,C,D</td>
<td>71</td>
</tr>
</tbody>
</table>
# List of Tables

2.1 VM overhead evaluation (packet delivery ratio) \hspace{1cm} 15

2.2 VM overhead evaluation (latency in seconds) \hspace{1cm} 16

2.3 AODV Parameters \hspace{1cm} 24

2.4 OLSR Parameters \hspace{1cm} 24

2.5 Experiment Rounds Summary \hspace{1cm} 26
I want to first thank Professor Mario Gerla for giving me the opportunity to work in a lab that is interesting and intellectually rewarding. I also would like to thank my fellow coworkers: Ian Ku, You Lu, Eduardo Cerqueira, Cathy Chen, Jaehan Lim, Xiao Li and many others who helped me out when I got stuck in my project. Without them this thesis would not have been possible.
Vita

2003  B.S. (Electrical Engineering), National Taiwan University, Taiwan
2005  M.S. (Electrical Engineering), National Taiwan University, Taiwan
2010  M.S. (Computer Science), UCLA, Los Angeles, California
2009–2012  Teaching Assistant, Computer Science, UCLA.
2012–2013  Research Assistant, Computer Science, UCLA.

Publications


Running Consistent, Parallel Experiments in Vehicular Environment Ian Ku, Jui-Ting Weng, Eugenio Giordano, Giovanni Pau, Mario Gerla, WONS, Bardonecchia,

Vehicular Testbeds - Model Validation before Large Scale Deployment Mario Gerla, Jui-Ting Weng, Eugenio Giordano, Giovanni Pau, ICNC’’, Maui, HI, January. 2012

xi
Pics-on-wheels: Photo surveillance in the vehicular cloud Mario Gerla, Jui-Ting Weng, Giovanni Pau ICNC 2013

Running Consistent, Parallel Experiments in Vehicular Environment Jui-Ting Weng, Ian Ku, Giovanni Pau, Mario Gerla, Journal of Communication 2013

Surveillance service on the open mobile cloud Jui-Ting Weng, Ian Ku, Mario Gerla WONS 2013

Reactive Pics-on-Wheel Service in Vehicular Cloud Jui-Ting Weng, Mario Gerla Technical Report
CHAPTER 1

Introduction

1.1 Toward Vehicular Service Cloud

As more and more mobile wireless devices are being used today, researchers are proposing location based services that exploit the collaboration of multiple end users. Rana et al [1] proposed to use cell phones to crowd source street noise. Google, Facebook, Yelp, and other social websites encourage users to perform location check-in to obtain more information about their current surroundings. On November 15, 2012, Google announced Ingress [2], a social game that encourages users to take pictures with their cell phone and upload them to the game center, which creates a data gold mine.

Most location aware crowd sourced applications use cell phone as client and rely on user interact with the application. However, another group of mobile computing devices are already deployed and run without much user awareness - vehicles. Today, in-car devices are mostly used to improve driving experience. However, people are beginning to make use of in car devices to build up applications. For example, Google applications crowd source traffic congestion based on GPS devices such as cell phones or tablets [3]. Cabspotting.org [4] tracks San Francisco’s taxicabs throughout the Bay Area. Vehicles gather and exchange peer to peer traffic information to support these mobile services and improve transportation intelligence.

As vehicles are getting smarter with more and more onboard processors and
devices, they are capable of handling more complex functionalities. In contrast to cell phones and tablets, in car computers have the power to provide real time, location based service without basically concern about battery lifetime. Recently, visionaries have proposed to connect vehicles and turn them into a cloud-like service provider, the Open Mobile Cloud. Gerla projected mobile cloud intelligent transport services in [5]. Olariu et al. proposed several architectures to tap vehicles for services as in a cloud in [6,7]. Huang proposed mobile software structures targeting these goals in [8]. Open mobile cloud targets the sharing of mobile hardware among peers in the network. Take for example the vehicular network; when people are driving, the sensor and processing capabilities of the on board computer are generally not fully utilized. These resources could be made available to remote users who wish to run location based services.

Based on the observation that the computer and hardware inside any modern vehicles are under utilized, this thesis proposed to collaborate multiple vehicles and make use of their spare computation resources. On the high-level view, the system looks like a service cloud which provides location aware services. The backend servers of such service are actually vehicles on the road.

While a single vehicle has the ability to provide location based services on its own, it does not provide enough interest for remote users to subscribe to its service. On the other hand, a group of vehicles, forming a vehicle cloud, provides a more widespread, consistent, and reliable service for the urban area. Using collaborative mobile vehicles, the vehicular cloud succeeds where the single vehicle cannot.

1.2 Cloud computing to Vehicular Cloud Service

Today, the term "cloud computing" is used to describe the concept to run remote service over the network. The backend are usually located in the data center and services can be provide to massive amount of users concurrently in a distributed
manner. The idea behind cloud computing is that computing, storage, and bandwidth are packaged as a service and can be utilized over a network, which is the Internet.

The idea of cloud computing is similar to an old concept time sharing. The time sharing mechanics can be traced back to 1960s. During that time, the only available computer is main frame computer which is only owned by government and corporations. Most people who cannot afford to have their own computer buy a share of the computing resource. Computer user submit their tasks to the mainframe through an input interface, and wait for the task to finish and print out. This client-server model allows many user to share the resource of the mainframe.

The vehicular cloud differs from todays Internet cloud in the nature of its hardware and sensor platform [7,8]. The vehicle cloud builds its platform opportunistically in a centralized or distributed control, from spare and idle resources made available by participating vehicles. Another major difference is the mobility nature of vehicular cloud. Vehicle cloud enables application to run on a particular location at a particular time, which is not possible for traditional cloud service. In fact, the majority of services provided by the vehicle cloud are expected to be location aware applications. In this thesis, we use two example to demonstrate how to deploy and design vehicular cloud with two examples - parallel experiments and surveillance service.

1.3 Parallel experiment platform with Vehicular Cloud

The dynamic nature of vehicular ad-hoc networks (VANETs) makes performance comparisons hard, because network conditions cannot be replicated. Base on the idea of time sharing, suppose multiple virtual machines are run in one Hardware provider. Each of these virtual machines (VMs) is a complete Linux experiment
environment. Moreover, these VMs experience the same wireless condition as they share the same physical network card.

Chapter 2 introduces Pep-Net (Parallel Experiment Platform for VANET), a VANET testbed where multiple experimental configurations run simultaneously on identical network conditions. PepNet exploits Xen and Gentoo to provide a virtualized environment at every node. Atop the virtualized environment, multiple virtual guests, each carrying an independent experiment, run in parallel sharing the same physical resources.

There are three benefits of using PepNet to run mobile experiments. (1) Virtual machines run various experiments simultaneously, so that each set of experiments encounters identical network conditions and thus produces consistent results. (2) Fewer physical machines are required. (3) Experiments are more consistent, easier to control, and the results are easier to interpret. To demonstrate the efficacy of PepNet, two well-known ad-hoc routing protocols, AODV and OLSR, are tested. Experiments confirm the results published in several previous studies, while the new testbed is more efficient and gives more consistent results.

1.4 Surveillance Service with Vehicular Cloud

The vehicular cloud reminds us of the Internet Cloud because of service provision and resource sharing. However, there are obviously major differences. First the vehicles today are virtually sensor platforms, and are the ideal probes of the urban environment [8]. Secondly, the vehicle cloud is formed not by design but opportunistically, from spare and idle resources made available by participating vehicles. The third major difference is mobility. Vehicular cloud exploits moving vehicles to perform services at different locations. In fact, the majority of services provided by the vehicle cloud are expected to be location aware services.
Mobile Vehicular Clouds are emerging as systems architectures suitable to provide services to both drivers and external customers. In this thesis we propose a surveillance service called Pics-on-Wheel. This service uses the Vehicular Cloud (with vehicles equipped with onboard navigators) and an Internet Cloud Server. The vehicles on board navigators periodically report their positions to the Server. The Server, upon receiving a photo request from a customer (either another vehicle or an external user) selects the proper client and orders the photo shot.

Mobile Clouds can be formed by pedestrians with smart phones. However, smart phones are limited by battery lifetime. Cellular phone users do contribute crowd-sourced images of exceptional events (e.g., riots, police abuses, natural disasters) for a civic sense of duty. However, pedestrian are less likely to agree to collectively offer crowd-sourcing as an organized service, in part because of the limited resources and also to avoid the risk of exposing themselves. Vehicles, on the other hand, have no power nor exposure concerns, thus they are more suitable to provide location aware services as a cloud.

In order to provide an organized service, we suggest mobile cloud to be managed by a centralized Cloud Service Provider. Cloud Service Provider acted as a broker between Service Customer, who actually request for location service, and the Hardware Provider, who owns the device to perform location service. Chapter 3 describes Pics-on-Wheel architecture for this service deployment and reports on a prototype implementation. A simulation study based on San Francisco taxi cab traces shows that the Pics-on-Wheel service yields latencies in the order of 5 minutes in well frequented areas even with a modest number of vehicles. The results also show that by introducing incentives for active driver participation, even minimal detours noticeably reduce latencies.

A surveillance vehicle cloud service may be used in several scenarios. For example, Google can lease and utilize cameras on board of other vehicles to provide enhanced real time street view updates and complement their fleet of Google street
view cars. The Department of Transportation can use the service to keep a better eye on all the traffic lights to check if they are all functioning. A regular driver can use the service to view the cause of traffic jam in the next block (say, a double parked van), or the parking space availability in the nearby streets. These are all possible usage of future surveillance related application, and we validate these ideas with our Pics-on-Wheel on-demand photography service prototype.
CHAPTER 2

PepNet: Running Consistent, Parallel Experiments in Vehicular Environment

2.1 Overview of Parallel experiments

Testbeds play a key role in ad hoc network research, because a wireless medium has physical characteristics that cannot easily be simulated. Over the years, many kinds of testbeds have been developed in ad hoc network. Performing testbed experiments in a mobile ad hoc network (MANET), however, is challenging.

The first challenge in a MANET testbed is the cost due to mobility. Devices require continuous power supply, human monitoring, and mobility support. Therefore, even a simple MANET experiment can be costly. In addition, when evaluating performance among protocols or applications, the cost of experiments increases proportionally to the number of different scenarios, which causes mobile experiments to be short and limited to a few rounds.

Another challenge for MANET testbed experiments is topology control. This becomes more difficult in a vehicular ad hoc network (VANET), where the topology changes are much faster than in a traditional MANET. Conventionally, vehicles are integrated with wireless devices to establish Vehicle-to-Vehicle (V2V) communication, and are driven around to provide mobility for the experiments [9–14]. Nevertheless, this can result in inconsistent environmental conditions. For example, if the objective is to compare two routing protocols, then common approach is to run experiments with routing protocols A and B separately. However, the
results are generally not fairly compared, because in ten minutes the external interference may have changed, the motion pattern of the various vehicles involved in the experiments may have changed (due to unpredictable traffic lights), and the radio propagation may have changed (say, due to mobile obstacles beyond experimental control).

Simulations provide an alternative approach to evaluate VANET research to solve the inconsistency problem in testbed experiments. Simulators such as ns-2, Opnet, and QualNet allow researchers to repeat experiments with different protocol configurations to ensure consistency. The whole system is under full control and thus one can adjust a few parameters while keeping the rest unchanged. However, simulators assume accurate modeling of physical characteristics such as mobility/traffic patterns, radio propagation models, and external interferences.

As to mobility patterns, since vehicles do not move randomly, it is unrealistic to use simplified mobility patterns such as random walk, random direction, or random waypoint. The most common approach is to collect traffic/mobility traces first and then run the simulations based on those traces. Traffic/mobility traces can be generated by traffic simulators, public transportation schedules, and real-time logs [15]. Such an approach allows one to evaluate the performance by simulations using complex and realistic mobility patterns.

Nevertheless, problems still remain in radio propagation models and external interferences. A common approach to simulate the radio propagation is to use well-established statistical models, which mostly do not account for static obstacles such as buildings and moving obstacles such as trucks. A model that estimates signal coverage through existing maps is available as in ref [16]. However, attenuation caused by dynamic factors such as moving obstacles is still difficult to reproduce. A naive solution, similar to what is often done for the mobility model, is to collect channel condition logs from real experiments. This is extremely costly since it records noise level and signal strength for all node pairs
every milli- or even micro-second throughout the experiment. A simplified version of this naive solution was proposed in ref [17], which records only channel connectivity. Nonetheless, connectivity itself is still not accurate enough to represent real channel conditions. These location-dependent and time-varying physical characteristics inevitably make simulations inaccurate.

Testbed approaches are the only way to conduct VANET experiments with high fidelity, but the dynamic nature of VANETs result in experimental inconsistency. Though it is possible to repeat enough experiments until the uncontrollable factors “even out”, this significantly increases the cost.

This chapter addresses these challenges by leveraging the resource sharing of virtual machines to perform parallel experiments. Its three major contributions are as follows. (1) Pepnet, a virtual machine based VANET testbed, compares different protocols under the same topology and channel conditions. (2) Virtual machines allow parallel experiments to be conducted with fewer physical machines. (3) The experimental results are easier to compare because of the experiment consistency. To demonstrate PepNet, two well-known ad-hoc routing protocols, AODV and OLSR, are tested. Our experiments confirm the results from several previous studies, while requiring fewer physical resources and providing more consistent results.

The rest of the chapter is organized as follows. Previous studies on testbeds and simulations are introduced in Section 2.2. Section 2.3 describes the platform details. Virtualization overhead is evaluated in Section 2.4. Comparisons between AODV and OLSR are reported in Section 2.5. Finally, conclusions and future extensions are presented in Section 2.6.
2.2 Related Work

VANET is an emerging technology that improves safety and provides comfort and convenience for vehicle drivers and passengers. Recently, many new protocols were proposed to address problems introduced by the new communication schemes that VANET enables, namely, Vehicle-to-Vehicle (V2V) communication and Vehicle to Infrastructure (V2I) communication. Routing is a particularly important issue since routes are changing all the time in both V2V and V2I communications. As a result, several routing protocols were proposed for different application needs [18–21]. However, no previous study provides a fair, accurate, and consistent evaluation.

The Wireless Signal Propagation Emulator developed by CMU [22] accurately emulates wireless signal propagation in a physical space. The emulator senses signals generated by known wireless sources through the antenna port, subjects the signals to the same effects that occur in a real physical space (e.g. attenuation, multi-path fading, etc.), and feeds the combined signals back into wireless cards. The emulator, however, has limitations in reproducing arbitrary motion patterns. In addition, although the propagation scenario is more realistic, it is still artificially created as opposed to real life measurement.

Orbit [23] is a testbed that combines an indoor radio grid emulator and an outdoor field trial network. This testbed is available for use either via remote or on site access. As for mobility support, the outdoor testbed is grounded, while the indoor emulator only supports virtual grid mobility. D. Rastogi et al. presented a comparison between AODV and OLSR, performed through the Orbit indoor testbed [24] showing that AODV performs better than OLSR in terms of stability. However, because the radio propagation model differs from the real world, Orbit cannot simulate wireless interferences.

The department of computer science at University Uppsala has opened to the
community the Ad hoc Protocol Evaluation Testbed (APE Testbed) [25]. APE is an encapsulated execution environment with tools for post test-run data analysis, similar to a small Linux package with ad hoc configuration and network traffic analysis tools. Lundgred et al. used APE to evaluate the performance of AODV and OLSR with up to 37 nodes along the indoor hallways and athletic fields in [26]. The result shows that AODV performs better than OLSR in this high mobility scenario. To the best of our knowledge, there is no VANET experiments use APE testbed.

Many academic facilities have mesh network testbeds that use AODV or OLSR to perform layer 3 routing. Some of the mesh testbeds are deployed in real environments, such as MIT RoofNet, Berlin Roof Net, and Mesh Networking from Microsoft Research [27–29]. These systems provide experimental results in real world channel conditions, but lack node mobility. Several vehicular experiments and testbeds have been proposed [9–14]. However, none of them compared performance among different routing protocols, because realistic mobility in vehicle experiments is hard to replicate.

To summarize, recent testbeds provide convincing channel conditions, but do not support repeatable, complicated mobility at the same time. In this chapter, a novel approach is used to perform parallel experiments. Thus the mobility and channel conditions do not need to be reproduced in order to compare different routing protocols.

2.3 PepNet Overview

When several experiments are performed at the same time in a single run, even if mobility patterns and channel conditions are not reproducible, these simultaneous experiments performed at the same time experience the same environmental conditions. The main objective of this study is to construct a platform for concurrent
experiments.

Our platform has two design goals. (1) Simple platform setup with commodity equipment, and (2) Direct protocol and application implementations in our platform without significant changes. A Parallel Experiment Platform for VANET (PepNet) is built to address these principles. PepNet has multiple virtual machines running on every mobile node. Each virtual machine runs one experimental configuration, and all virtual machines at a mobile node share hardware resources and have identical mobility. In additions, each virtual machine fulfills the system requirement for one experiment. The following sections first describe the hardware platform and subsequently the software setup details.

2.3.1 Hardware Platform

Figure 2.1: Node setup on a vehicle

PepNet nodes are common commercial laptops with an Intel Core 2 Duo CPU, 2GB of RAM and a 120GB hard drive. Each laptop is instrumented with a Ubiquiti SRC wireless card with Atheros 802.11 wifi chipsets (AR5004). The Atheros 802.11 wifi chipset is supported by the open source Linux madwifi driver [30], which allows many customized settings including fixed channel selection, transmission power adjustment, and monitor mode support. For our experiments, all
physical wireless cards are in ad hoc mode, using channel 1 only. The transmission
time power is set to the hardware supported maximum (19dBm). The wireless card is
connected to a magnetic mount antenna with an 8dB nominal gain. Each laptop
is also equipped with a GPS receiver to track the positions of the nodes during
the experiments. Figure 2.1 shows an example of a node setup on a vehicle.

2.3.2 Software Platform

Each PepNet node runs the Linux Gentoo distribution (kernel version 2.6.21)
patched with Xen [31,32]. Xen is an open source industry standard virtualization
environment that allows several virtual machines (Xen guests) to share hardware
as shown in Fig. 2.2. One virtual network interface card (eth0) in every guest
operating system and the physical wireless card in the host operating system
(Xen host) are connected together through a Linux virtual bridge. This bridge
handles all incoming and outgoing wireless traffic, so Xen guests route as if they
are directly connected to the wireless network interface. An advantage of the
bridging approach is that Xen host needs not know what routing protocols are
run in the Xen guest environment.

In addition to the first bridge, a second bridge allows the communication be-
tween host and guest without interfering with experimental traffic. A preliminary
experiments found that time drifts between the Xen host and guests even with the synchronization that Xen provided. For better time synchronization, the second bridge enables the Network Time Protocol (NTP) to correct for the time drift between Xen host and guests clocks. The second bridge also passes geographical information to the Xen guests. The Xen host is connected to the GPS device by gpsd [33], an open source daemon that provides a network socket interface for retrieving location information. Through the Ethernet bridge, Xen guests can access GPS information via gpsd. This allows the use of applications and protocols that require GPS information.

2.4 PepNet Overhead Evaluation

Virtualization introduces two types of additional overhead, which may effect experiment results:

- **Virtualization Overhead**: Virtualization adds an extra software processing layer between application and hardware. In addition, hardware resources are used to run the virtual machines. Therefore, system performance is worse than a regular Linux system.

- **Sharing Overhead**: When multiple virtual machines transmit at the same time, they contend for the same physical hardware. This contention lowers the maximum throughput the hardware can achieve and increases the packet drop rate.

This section quantifies the "safe zone" of the PepNet such that the overhead does not significantly affect the validity of the experiment results.
Figure 2.3: Virtualization Overhead experiment scenarios. The upper part shows the packet delivery environment from one Xen guest to another Xen guest on another physical machine. The bottom part shows the source and destination of each scenario, and the processing performance at each interface.

Table 2.1: VM overhead evaluation (packet delivery ratio).

<table>
<thead>
<tr>
<th>Checkpoint Scenario</th>
<th>Sender</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (10K)</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>2 (10K)</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
<td>0.995</td>
</tr>
<tr>
<td>3 (10K)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.646</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.646</td>
</tr>
<tr>
<td>4 (10K)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.617</td>
<td>0.617</td>
<td>0.617</td>
<td>0.617</td>
<td>0.617</td>
</tr>
<tr>
<td>1 (100K)</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>2 (100K)</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>0.976</td>
<td>0.955</td>
<td>0.974</td>
<td>0.988</td>
<td>0.988</td>
</tr>
<tr>
<td>3 (100K)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.653</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.653</td>
</tr>
<tr>
<td>4 (100K)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.655</td>
<td>0.646</td>
<td>0.645</td>
<td>0.649</td>
<td>0.648</td>
</tr>
</tbody>
</table>
Table 2.2: VM overhead evaluation (latency in seconds).

<table>
<thead>
<tr>
<th>Checkpoint Scenario</th>
<th>Sender</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (10K)</td>
<td>4.59</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4.59</td>
<td>4.62</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4.62</td>
</tr>
<tr>
<td>2 (10K)</td>
<td>4.68</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4.67</td>
<td>4.72</td>
<td>4.72</td>
<td>4.72</td>
<td>4.72</td>
<td>4.72</td>
</tr>
<tr>
<td>3 (10K)</td>
<td>4.67</td>
<td>4.62</td>
<td>4.62</td>
<td>4.62</td>
<td>4.62</td>
<td>4.79</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>4.81</td>
</tr>
<tr>
<td>4 (10K)</td>
<td>4.43</td>
<td>4.43</td>
<td>4.43</td>
<td>4.42</td>
<td>4.42</td>
<td>4.50</td>
<td>4.503</td>
<td>4.503</td>
<td>4.503</td>
<td>4.60</td>
</tr>
<tr>
<td>1 (100K)</td>
<td>51.78</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>51.78</td>
<td>51.83</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>51.83</td>
</tr>
<tr>
<td>2 (100K)</td>
<td>54.89</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>54.89</td>
<td>55.05</td>
<td>55.05</td>
<td>55.05</td>
<td>55.05</td>
<td>60.05</td>
</tr>
<tr>
<td>3 (100K)</td>
<td>49.48</td>
<td>49.48</td>
<td>49.48</td>
<td>49.48</td>
<td>49.48</td>
<td>51.28</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>51.26</td>
</tr>
<tr>
<td>4 (100K)</td>
<td>54.71</td>
<td>54.70</td>
<td>54.70</td>
<td>54.70</td>
<td>54.70</td>
<td>54.75</td>
<td>54.75</td>
<td>54.75</td>
<td>54.75</td>
<td>54.75</td>
</tr>
</tbody>
</table>

2.4.1 Virtualization Overhead

Figure 2.3 shows the end-to-end packet delivery process on the Xen virtual machine. After the physical wireless interface at the Xen host receives the packet, it is forwarded to a virtual interface via the virtual bridge at the Xen host. Then, every packet received at this virtual interface is copied to the destination Xen guest. Similarly, outgoing packets go through the reverse process and have additional processing overhead in the virtual environment.

The bottleneck of our system was examined by considering four experiments: (1) Linux (Xen host) to Linux, (2) Linux to Xen guest, (3) Xen guest to Linux, and (4) Xen guest to Xen guest. The Linux system here refers to the Xen host OS without running virtual machines in it. In every scenario, the sender generates and sends to the receiver burst traffic of 10k and 100k UDP packets, each containing 1000 bytes payload. For every set of the scenarios, a "checkpoint" is set at every interface and both UDP sender/receiver applications. The experiment results are
shown in Table 2.1 and Table 2.2.

Each column represents a different checkpoint, and each row represents the combination of the scenario and the traffic load. The (10K) and (100K) tags in each row represent the number of packets generated. Table 2.1 shows the packet delivery ratio, which is the number of packets received at a checkpoint divided by the number of packets generated at the sender. Table 2.2 is the interval between the first packet and the last packet received at every checkpoint in seconds. These numbers were collected by running the packet sniffer tshark [34] at every interface.

Comparing the packet delivery ratio sequentially at all interfaces in Table 2.1, there is a steep decrease between checkpoint D and E in scenarios 3 and 4, where the packets originate from the Xen guest instead of the Xen host. On the contrary, when packets originate from Xen host in scenarios 1 and 2, packets entering the outgoing interface D are successfully transmitted to checkpoint E with high probability. The transmission between checkpoint D and E is carried out by two 802.11 wireless cards. This shows that when bursty packets are generated on the same operating system as the physical interface, the Xen host has better scheduling to handle packets, so fewer packets are dropped at the wireless device.

Next, the reliability of the Xen system is considered. Inside the sending machine, there is no packet loss at any checkpoint, as shown in scenario 3 and 4. In addition, a test between the Xen host and guest using iperf shows the bandwidth between Xen host and guests are more than 8Gb/s, which is significantly larger than the wireless bandwidth. Therefore, the Xen host and guest communication in the sending node is 100% reliable.

On the other hand, some packet loss is found inside the receiving machine, in scenario 2 and 4 with 100K bursty traffic. However, the total number of receiving packets at each interface is not always decreasing. From row scenario 2 (100K) and scenario 4 (100K), columns G and H, more packets are received at the last interface than the virtual bridge. This contradictory result implies that tshark
is likely not 100% accurate. The experiments show that tshark has at least 4% error, as some packets that are not sniffed at one interface appear later in another interface. Although this complicates the result, packet counts sniffed at every interface inside the receiving node are still close to each other. This suggests that after packets are received at the physical interface of the receiving node, they are delivered to the application with a high probability.

Table 2.2 shows the interval between the first packet and the last packet sniffed at every interface. This number remains fairly constant among all interfaces for every scenario. Because all interfaces besides the two wireless devices have little packet loss, the processing time at each virtual interface does not contribute much to the delay, according to queueing theory. It also implies that the next bottleneck besides wireless devices processing in the experiments is the UDP packet generator. In addition, scenario 1 on row 1 and row 5 represents the performance of the native Linux system, which shows the network interface card achieved 17.8Mb/s while transmitting 10k packets, and 15.4Mb/s while transmitting 100k packets.

Finally, different loads of UDP Constant Bit Rate (CBR) traffic is used to examine the threshold of the system in order to find the ”safe zone” to perform experiments. The CBR traffic is generated by the UDP application with 1000-byte packets with a fixed interval instead of bursty traffic. The network configuration is the same as scenario 4: traffic generated from one Xen guest and delivered to another Xen guest on a different node. The results are shown in Figure 2.4. The outgoing interface can handle at least 10Mb/s of traffic without introducing massive end-to-end packet loss. We conclude that the Xen system is reliable when the wireless traffic load is under 10Mb/s.
2.4.2 Sharing Overhead

On our virtualized platform, multiple Xen guests share physical resources. It is therefore important to understand the maximum network load at which the sharing causes performance drops or excessive delays. In our setup, the Xen host connects multiple Xen guests to the physical network interface through a Linux network bridge. The operating system provides a share of CPU time to the virtual machines, and thus to the read and write locks on the network bridge. The bridge queue policy is First In First Out (FIFO) coupled with the fair share of the locks that avoids starvation and provides fair sharing of the resources. This section exams the capacity when multiple virtual machines are run at a single node.

Two sets of experiments were performed: (1) many-to-one, and (2) one-to-many. The experimental traffic is generated from Xen guest(s) in one machine and deliver to other Xen guest(s) at another machine. For the many-to-one scenario, different numbers of Xen guests are run on a single Xen sending machine. They

Figure 2.4: Overhead introduced by Xen virtualization: packet delivery ratio while performing the VM to VM experiment with different loads of CBR traffic.
simultaneously send UDP traffic to a single Xen guest at the receiving machine. In the one-to-many experiments, multiple applications run together at a single Xen guest, and each sends UDP traffic to different receiving Xen guests, while all receiving Xen guests are located at a single Xen receiving machine. Every Xen guest run the same CBR traffic in every experiments. Each traffic is composed of 1000-byte payload UDP packets with a fixed interval determined by the aggregated throughput. Experiments are finished when an aggregated 100k packets are sent.

The first goal in this section is to understand how the number of Xen guests affects the network capacity. Figure 2.5 shows the packet delivery ratio with different numbers of Xen guests in both scenarios. The overall packet delivery ratios among different numbers of Xen guests are almost identical. This shows that the system can handle at least five concurrent virtual machines without introducing significant overhead. Moreover, in the one-to-many experiments, the packet delivery ratio slightly increases as the number of virtual machine increases. It is likely because running multiple applications slightly increases the packet generation time. Therefore, the physical devices have longer time to handle each packet and in turn achieve higher delivery ratios.

Another important issue is the fairness among the virtual systems when the network is fully loaded. Even though the number of virtual machine does not affects the aggregated throughput, the network resource may not be shared fairly. Figure 2.6 demonstrates the case of five concurrent virtual machines, where each line represents a different UDP flow. The packet delivery ratio for each UDP flow is defined as the packets received at the destination application layer divided by the packets sent. When the traffic is under 10Mb/s, every sender attains close to 100% delivery. As the aggregate throughput increases, virtual machines start to contend for physical resources. Consequently, all virtual machines suffer from similar percentage of packet loss. The maximum difference of packet delivery ratio among all UDP flows in many-to-one experiments is 11%, where as the ratio is
Figure 2.5: Overhead introduced by Xen virtualization: packet delivery ratio as a function of UDP CBR for multiple UDP flows generated at different virtual machines.
6% in the one-to-many experiments. This shows that Xen prevents a single Xen
guest from dominating the networking device when contention happens.

To sum up, multiple Xen guests can utilize the bandwidth well if aggregated
throughput is under 10Mb/s. When congestion happens, all Xen guests share the
bandwidth with reasonable fairness.

2.5 Field Experiments

PepNet is used to perform comparisons between two well known ad hoc routing
protocols, AODV and OLSR. We varied the use of AODV and OLSR (singularly or
concurrently), interference nodes and the number of mobile nodes to perform eight
rounds of 20-minute experiments. Table 2.5 reports the configuration for each
experiment. Section 2.5.1 presents the details of the setup for each experiment
round, and Section 2.5.2 presents the comparison result between two protocols.
AODV is a reactive protocol, which allows it to perform better route selection than
OLSR in a frequently changing topology [35]. This has been verified by numerous
simulations and testbeds with limited mobility support. PepNet achieves the same
conclusion in real world traffic, with fewer experimental resources.

2.5.1 Experimental Setup

The experiments were performed around the UCLA Engineering IV building with
eight different network configurations described in the following subsections.

2.5.1.1 Xen Host Setup

Mactrace Each Xen host ran a mactrace tool that broadcasts a hello beacon
every 200 ms, which contains position and timestamp information. These hello
beacons were used to construct the network connectivity matrix over time for best
Figure 2.6: Xen fairness evaluation: packet delivery ratio for different Xen guest pair while running 5 VMs in parallel
route investigations.

**Tshark**  Each Xen host ran tshark to log all packet sniffed at the wireless interface.

**Synchronization**  All the systems are synchronized via NTP. This allows temporal correlation among the trace logs with the precision of ten macro-seconds.

<table>
<thead>
<tr>
<th>Hello message interval</th>
<th>500ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowed hello loss</td>
<td>2</td>
</tr>
<tr>
<td>Delete period</td>
<td>1s</td>
</tr>
<tr>
<td>Active route timeout</td>
<td>2s</td>
</tr>
</tbody>
</table>

Table 2.3: AODV Parameters

<table>
<thead>
<tr>
<th>Hello message interval</th>
<th>500ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello message validity time</td>
<td>1s</td>
</tr>
<tr>
<td>Topology control (TC) message interval</td>
<td>1s</td>
</tr>
<tr>
<td>Topology control (TC) message validity</td>
<td>2s</td>
</tr>
</tbody>
</table>

Table 2.4: OLSR Parameters

**2.5.1.2 Xen Guest Setup**

**Routing Protocols**  Each node, ran two Xen guests in addition to the Xen host. One Xen guest ran an implementation of the ad hoc on demand distance vector routing (AODV) AODV-UU [36,37], and the other Xen guest used the Optimized Link State Routing (OLSR) [38,39]. Tables 2.3 and 2.4 summarize the parameters
of AODV and OLSR. These variables are based on the previous work in [24] and account for the vehicular scenario used in the experiments.

**Network Traffic** A simple application generated a constant UDP stream of 50-byte data segments every 150ms. Each packet contained a packet sequence number and the timestamp when the packet was created. The traffic source and destination pairs are described in the next paragraph.

### 2.5.1.3 Topology and Interference Setup

Figure 2.7 shows the bird’s-eye view of the physical topology setup. Four fixed nodes remained at the same place for all eight experiments. The antennas of these nodes were placed on top of a 1.5 meter stand to ensure good signal propagation. The mobile nodes were placed on vehicles and their antenna was placed on the top of the cars to avoid interference due to the automobile body. The four fixed nodes, represented by the pin in Figure 2.7, were placed at the four corners of the building. The resulting rectangle was approximately 60 meters wide (East-West) and approximately 50 meters long (North-South). Each fixed node was in line of sight with the nodes placed at neighboring corners. Therefore, neighbor nodes were connected and diagonal connections are blocked by the building.

Half of the experiments used two interference nodes indicated by the wave icon in Figure 2.7. These interference nodes generated layer 2 bursty broadcast traffic of random length uniformly distributed from 0 to 100 packets of random size uniformly distributed from 50 to 1000 bytes. After each burst, they idled for a random period of time uniformly distributed in the interval of [0,30] seconds.
2.5.1.4 Experiment Traffic Setup

The summary of the 8 rounds of experiments is reported in table 2.5. AODV alone was used in rounds 1 and 3; OLSR alone in was used in rounds 2 and 4. Both AODV and OLSR were used in parallel in rounds 5 through 8. Traffic generated from the Xen guest running AODV were delivered to the Xen guest which also ran AODV at the receiving machine, and similarly with to OLSR.

In the single mobile node scenarios, the test traffic was generated from the mobile node and delivered to one of the four fixed nodes. After sending the packet for two minutes, the UDP traffic paused for one minute, and then repeated the process to another fixed node in a round-robin manner. In the two mobile node scenario, both mobile nodes moved clockwise with different speeds. When two mobile nodes met, the faster mobile node overtook the slower one. The faster mobile node sent UDP stream to the slower mobile node for two minutes, paused for one minute, and then repeated the same process. Each round lasted for 20 minutes, which contained seven 2-minute sub-sessions.

Table 2.5: Experiment Rounds Summary

<table>
<thead>
<tr>
<th>Experiment Name</th>
<th>AODV</th>
<th>OLSR</th>
<th>Interference</th>
<th>Mobile Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>√</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Round 2</td>
<td></td>
<td>√</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Round 3</td>
<td>√</td>
<td></td>
<td>√</td>
<td>1</td>
</tr>
<tr>
<td>Round 4</td>
<td></td>
<td>√</td>
<td>√</td>
<td>1</td>
</tr>
<tr>
<td>Round 5</td>
<td>√</td>
<td>√</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Round 6</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>1</td>
</tr>
<tr>
<td>Round 7</td>
<td>√</td>
<td></td>
<td>√</td>
<td>2</td>
</tr>
<tr>
<td>Round 8</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>2</td>
</tr>
</tbody>
</table>
2.5.2 Experiment Results

This section covers three areas: (1) the mactrace tool showed the dynamics of the VANET experiment environment, (2) sequential experiments with AODV and OLSR did not provide consistent performance comparisons, and (3) parallel experiments consistently provided comparable results.

2.5.2.1 VANET Dynamics

The network topology of the experiment is reflected by the optimal hop count, which is defined as the shortest hop count from one node to another. During the experiment, mactrace periodically broadcasted beacon packets. Every other node received the beacon packet and recorded the existence of a one-way connectivity at that time. Using the mactrace log, the network connectivity matrix was constructed, and the shortest hop counts between any two nodes at any given time
was obtained by running a shortest path algorithm.

Figure 2.8 shows the optimal hop count from the moving node to one fixed node in round 1 and round 2 for 10 minutes. Because the four fixed nodes formed a complete cycle, the theoretical optimal hop count is 2 hops. However, some uncontrollable factors such as other cars blocking the signal line of sight or random wireless noises interfered with the optimal route resulting in unpredictable behavior. According to Figure 2.8, the optimal hop count repeatedly alternated between 1 and 2, and occasionally reached 3 hops in both round 1 and round 2, the single-mobile-node scenarios. The result shows network topology looks considerably steady in these rounds.

The two-mobile-node scenarios were more complicated, and the optimal hop count from the faster mobile node to the slower mobile node is shown in Figure 2.9. To quantify the difference between two rounds, 64% of the time in round 7 the optimal hop count was 1, while only 45% of the time in round 8 the optimal hop count was 1. Moreover, the topology change in round 8 happened more frequently than round 7. The optimal hop count changed 58 times and 84 times in round 7 and round 8, respectively. If we consider a successful hop count change to be the one that lasts at least 5 seconds, then round 7 and round 8 contained 6 and 13 successful hop count changes, respectively. This shows that even in a simple experiment environment, – two cars circling four fixed nodes around a building – it is still difficult to control the network topology.

2.5.2.2 Sequential Experiments

The experiment in round 1 through 4 study the performance comparison between AODV and OLSR, with and without the artificial interference signal. Experiment rounds 1 through 4 were run sequentially, and their results are shown in Figure
Figure 2.8: Network connectivity: optimal hop count from mobile sender to one fixed node over time for experiment round 1, 2
Figure 2.9: Network connectivity: optimal hop count from mobile sender to another mobile receiver over time for experiment round 7, 8
2.10. The performance metrics used here are the packet hop count distribution and the packet delivery ratio. The packet hop count is the number of hops that a successfully received packet takes to reach the destination. The packet delivery ratio is the total number of packets received divided by the total number sent.

Figures 2.10a and 2.10c show the packet hop count distribution. In general, protocols with a better route selection have a smaller hop count because they react to topology changes faster. Figure 2.10a shows that the OLSR performed slightly better in hop count selection than AODV in a 20-minute experiment. On
the other hand, 2.10c draws the opposite conclusion. The packet delivery ratio comparison for the first four rounds is shown in Figures 2.10b and 2.10d. AODV had a better packet delivery ratio than OLSR, although in the first two rounds AODV performed 10% better and in the next two rounds AODV performed only 1% better. Therefore, even with a considerably controlled mobility (one mobile node circling four fixed nodes), the environmental changes still increased experimental dynamics. Thus, running single protocol experiments (no parallelization) for the purpose of evaluating alternate protocol was misleading. A traditional way to solve this problem is to “even out” the external influences by repeating the experiments multiple times with longer durations. However, performing real VANET experiments is costly.

2.5.2.3 Parallel Experiments

This section shows that the experimental results of the four parallelized experiments provided better consistency than the sequential experiments.

Figure 2.11 shows the results of the parallel experiments in the single-mobile node scenario, as in rounds 5 and 6. Figures 2.11a and 2.11c present the packet hop count distribution for both AODV and OLSR in experiments rounds 5 and 6, and Figures 2.11c and 2.11d show the packet delivery ratio. We note that the AODV consistently outperformed OLSR in the single mobile node scenario because AODV exhibited better hop count selection and higher packet delivery ratio in both rounds. Moreover, this conclusion can be safely drawn only because both protocols experienced the same environment condition so the dynamic environment factors were eliminated.

Next, consider the two-mobile-node scenarios that have uncontrollable connectivity patterns, as shown in Figure 2.8b. The result in Figure 2.12 shows parallelized experiments still provided consistent results. Figure 2.12 shows the
Figure 2.11: Single mobile node parallel experiments: rounds 5 and 6
packet hop count distribution and the packet delivery ratio for both protocols in rounds 7 and 8. AODV still consistently outperformed OLSR both in terms of the hop count and the packet delivery ratio. This result shows parallel experiments delivered accurate result in a single experiment run even when the mobility pattern is uncontrolled.

Finally, a part of the parallelized experiments show that performing sequential experiments may lead to false conclusions. According to Figure 2.13a, the environment in round 8 was noisier than round 7, since the packet delivery ratio
dropped by 15%. Suppose only protocol OLSR were evaluated in round 7, and only AODV were evaluated in round 8; then, after two 20-minute of experiments, OLSR would have outperformed AODV in both the packet delivery ratio and the hop selection, which would be a false conclusion, as shown in Figure 2.13. This shows sequential experiments do not guarantee a controlled experiment environment, and parallelized experiments battle the uncontrollable experimental factors efficiently.

2.6 Conclusion and Future Work

Running experiments in testbeds is essential to compare different protocols in realistic scenarios. However, the dynamic nature of VANET makes it difficult to faithfully reproduce the experiment environments. The brute force solution is to repeat the experiment over and over again in hope that the external interferences and obstacles effects are "evened out". However, repeating experiments in a vehicle testbed is very costly and some scenarios are difficult to precisely reproduced.
One possible approach to perform parallelized experiment is by installing multiple wireless nodes (e.g. laptops) in a single vehicle. The drawbacks for this approach are (1) the hardware cost increases proportionally to the number of parallelized scenarios and (2) the coordination between different hardware becomes more complicated. The virtual machine, parallelized testbed, we proposed removes the aforementioned limitations by performing parallel runs under the same environmental conditions with a single set of hardware. The virtual machine also provides a complete operating system, so protocols with different system requirement can still be processed in parallel. To sum up, the virtual machine parallelization provides consistent and reliable results from a single run with lower cost.

A new virtual machine based testbed, PepNet, addresses the inconsistency issues in VANET performance evaluation. By running experiments concurrently, protocols or applications encounter identical topology, channel conditions, and external interferences, thus avoiding difficulties in reproducing identical environments. Besides, there are two additional benefits of running parallel experiments in virtual environment. First, virtual machines provide a complete operating system so protocol implementations with different system requirements can be run in parallel. Second, experiments are performed with a single set of hardware, leading to greatly reduced hardware cost. These benefits make virtualization a cost efficient way to conduct VANET experiments.

Another contribution of this topic is to understand the virtualization overhead so that experiments can be designed accordingly. Section 2.4 shows that Xen can handle 10Mb/s traffic reliably. In addition, the checkpoint experiments show that the bottleneck is the wireless interface card located at the sender node. Our experiments also found the tshark error at the receiving virtual machine, which makes it hard to quantify the receiving node error rate. Experimental results also
show that when aggregate throughput does not exceed the threshold, Xen supports up to 5 parallel virtual machines without introducing significant overhead. It is our next goal to seek a solution for the high packet loss rate to improve the performance of virtual machines.

Finally, several field experiments demonstrated the importance of parallel experiments. With only two mobile nodes, the network topology is already quite complex and difficult to reproduce. Sequential experiments in a simple scenario do not provide a consistent comparison result, while parallel experiments provide consistency at all times.
CHAPTER 3

Pics-On-Wheel: Surveillance Service in the Vehicular Cloud

3.1 Overview

Modern vehicles are capable of a wide range of applications. We introduce surveillance as one example of location-based service provided by the vehicular cloud. The surveillance application exploits the mobility of the service nodes (vehicles) to extend the coverage beyond the reach of static sensors (e.g., video cameras).

While a single vehicle has the ability to provide location-based services on its own, it does not provide enough interest for remote users to subscribe to its service. On the other hand, a group of vehicles, forming a vehicle cloud, provides a more widespread, consistent, and reliable service for the urban area. Using collaborative mobile vehicles, the vehicular cloud succeeds where the single vehicles cannot.

Mobile Clouds can be formed also with smart phones. However, smart phones are limited by battery lifetime. Cellular phone users do contribute crowd-sourced images of exceptional events (e.g., riots, police abuses, natural disasters) for a civic sense of duty. However, they are not likely to agree to collectively offer crowd-sourcing as an organized service, in part because of the limited resources and privacy issues. Vehicles, on the other hand, have no power nor exposure concerns, thus they are a more suitable solution to form a mobile cloud that provides surveillance services.
In order to provide an organized service, we suggest mobile cloud to be managed by a centralized Cloud Service Provider. A Cloud Service Provider acted as a broker between Service Customers, who actually request for location service, and the Hardware Provider, who owns the device to perform location service. Figure 3.1 shows the hierarchy of the three roles in a mobile cloud system, which is illustrated by our Pics-on-Wheel system as an example.

The Pics-on-Wheels service selects a group of vehicles to take photos of a given urban landscape within a given time, as requested by a Service Customer. To participate in Pics-on-Wheels, Hardware Provider (the vehicle) registers to the centralized Cloud Service Providers as opt-ins. The vehicles upload their GPS location periodically to the Cloud Service Providers. The on-board navigation system in each vehicle maintains the trace of the vehicle for a predefined time period. In the next chapter, we explain how Pics-on-Wheels service works.

![Figure 3.1: Mobile Cloud System](image)

### 3.2 Pics-On-Wheel Service

#### 3.2.1 Terms definition

First, we define the term Service Target. Service Target is provided by the Service Customers, which contains Service Time and Service Area as the time and location in which the service needs to be completed. Since vehicular cloud is focusing on
location-based service, it defines where and when the physical hardware is required to present in order to provide service. An example of Service Target would be during 2:00pm ~2:10 pm at 3rd street. Second, the term Job Notification is a message delivered by Cloud Service Providers to a Hardware Provider informing a Service Target sent by Service Customers.

Once the Service Request reaches the Cloud Service Provider, the immediate problem for Cloud Service Provider is to decide who is capable to perform service based on hardware, time, and location. Current in-car GPS receives the signals from the satellites to compute for its position locally. Therefore, the location information needs to be delivered to the Cloud Service Provider for scheduling service. The traffic overhead can be unmanageable when thousands of vehicles must update their positions within seconds. In fact, the authors of RoadTrack [40] suggest that vehicle updates should be no faster than several minutes per vehicle, lest the spectrum becomes congested. In order to solve this problem, we discuss two different scenarios: (1) No route information (2) vehicle given route information in the next section.

### 3.2.2 Finding Hardware Provider without route information

Consider a basic mobile cloud use case. Cloud service Provider maintains the position of Hardware Provider. After receiving the requests from the Service Customer (police, paramedics, researchers, etc.), the Cloud Service Provider attempts to forward the Job Notification based on the best hardware location information. If the Cloud Service Provider has instant location access to each Hardware provider, Job Notification would only be sent to vehicles driving towards the Service Target. All vehicles physically able to perform service are being selected, and only these vehicles are being notified. However, the most recent location information is kept inside the vehicles as discussed earlier. Moreover, a vehicle traveling 45 miles per hour is equal to 20 meters per second. For location application which has a
target location up to miles, such as pollution sensing, the error caused by location is less critical. For application requiring accurate position, such as Surveillance or Wi-Fi communication, it is possible to miss a hardware provider which could have finished the service because the Cloud Service Provider lacks accurate position information.

There are two extremes ways that the Cloud Service Provider perform Service Request Notifications. It can notify only vehicles which happen to perform location update in the Service Target (most efficient), or it can notify all vehicles around Service Target to make sure that every vehicle entering the Service Target is notified. We define the zone where vehicles are able to reach Service Area within the Service Time as the "Zone of Service". A vehicle at the boundary of Zone of Service can reach Service Area at the end of Service Time if it travels most efficiently. In other words, Zone of Service can be precomputed by the Cloud Service Provider based on the ending of Service Time, current time, and traffic speed. The area Zone of Service changes with time. At any time, Zone of Service contains all possible vehicles able to perform the service, as shown in Figure 3.2. As the Service Time ends (current time approaching the end of the Service Time) the area of Zone of Service shrinks to the size of the Service Area. Since vehicles outside the Zone of Service have no chance of reaching Service Target, Zone of Service provide a bounded region of which Cloud Service Provider would send the Job Notifications.

The Zone of Service can be used to rule out vehicles that have no chance entering the Service Target. If the Cloud Service Provider sends out Job Notifications to every vehicle reporting its location inside the Zone of Service, every vehicle passing the Service Target receives Job Notifications. However, when the Service Time last for a certain period, the size of Zone of Service would increase squarely. Moreover, not all vehicles in zone of service are actually going to drive past the Service Area before they leave the Zone of Service.
The same reliability can be achieved with less Job Notifications if vehicles are guaranteed to update frequently. If the interval between two consecutive vehicle location update is less than 60 seconds, it must perform more than one location update in the area that takes less than 60 seconds to drive to Service Area. Intuitively, this means Job Notification does not need to be sent unless the vehicle is close enough to the Service Area, where the distance is determined by the location update frequency.

To sum up, when there is no routing information provided by the vehicle, the most efficient way to distribute Job Notification is to only target vehicles already in the Service Target. On the contrary, the most reliable way is to distribute Job Notification to vehicles that are bounded by both Zone of Service and location update intervals.
3.2.3 Finding Hardware Provider with route information

In the previous section, we assume the Cloud Service Provider have no information on the route of each vehicle. If the route of one vehicle is known to drive past the Service Target, sending out notification of location service becomes a trivial task - V Send Job notification to vehicles driving to Service Target, and vice versa. Moreover, the Zone of Service requirement in the previous section still needs to hold for a vehicle to even be considered. Thus, the difficulty of this scenario is to inform Cloud Service Provider the route of a vehicle.

We propose a scalable location management scheme by combining the location update system with navigation system as described in [5]. The navigation system provides a turn-by-turn navigation to driver as it is today, but the computation is done in data center rather than a in-car device. Therefore, the navigation system can manage traffic flow and provide better route optimization in a crowded traffic scenario.

In the mobile cloud, we assume the vehicle follows the instruction of navigation system. Vehicles periodically exchange their current location for the driving instruction from the navigation system in the data center. The navigation system is a centralized service which takes driving direction, vehicle traffic optimization and location services into account to provide real time driving route update through 3/4G connection.

3.3 Photo Service in Vehicular Cloud

In this section we describe two types of surveillance services using the vehicle cloud structure we proposed: Photo Shooting Service and Forensic Picture Service. In general, the Pics-on-Wheels service uses a group of vehicles to provide photo shots of a given urban landscape within a given time interval as requested by a customer.
When the request time interval is in the future, we called it the Photo Shooting Service. When the request time interval is in the past, which means we are asking for a picture taken by any vehicle before the current time, we name it Forensic Picture Service. To participate in this Service, vehicles register to the centralized cloud manager as opt-ins. They also upload their own GPS location periodically to the cloud manager. In different scenario, the on board navigation system may also provide driving directions for vehicles to follow.

3.3.1 Photo Shooting Service

The vehicle selection procedure for the photo shot can be summarized in four steps: Step 1: The Cloud Service Provider upon receiving the request from a customer computes ZoS. Step 2: The Cloud Service Provider estimates the number of qualified candidates in ZoS and accepts/rejects the request from the Customer. Step 3a (w/o Navigation System): If accepted, the Cloud Service Provider invites vehicles in ZoS to provide the service. Step 3b (Navigation System): If accepted, the Cloud Service Provider invites vehicles passing Service Target to provide the service. Step 4: The vehicle accepts/rejects the invitation.

In this study we assume that the Pics-on-Wheel service is centrally controlled via an Internet Server. The Server, vehicles and customers communicate via 3G network. For example, the taxicabs communicate with the dispatcher (once every few minutes, say). In the same way they also talk (via an interactive on board navigator) to the Navigation Server to get a better route. Since the 3G or LTE [41,42] system does not support broadcast from an Internet Server to cellular users, the Cloud Service Provider must initiate one connection for every vehicle it wishes to be connected.

It is thus clear that one of the main performance factors of this service is the frequency in which the vehicles and Cloud Service Provider communicate. The en-
suing delay is a penalty of the 3G-connection model with respect to the broadcast model. There are, however, two immediate advantages. The centralized Server can screen vehicle requests for sensitive services, before the service is announced to the entire vehicle cloud. Also, the Server, in both navigator and dispatcher case, learns the final destination of the vehicle and can infer the path to such destination.

The next step is to find the candidates that can perform the service. These candidates by definition are in the ZoS. Note that only a subset of vehicles in this region can perform the job, namely the ones that will pass by the target. With the ZoS knowledge, the Server can estimate how long it will take to satisfy a customer request. Assuming no prior knowledge of vehicle trajectories and no Navigation System, the Server can compute the probability $P$ that a randomly chosen vehicle within the circle drives by the central target, under the assumption of uniform vehicle direction selection. Knowing $P$, the average urban traffic density, and query frequency, the Server can estimate the success rate, and thus relay this information to the perspective customer. For example, if the customer wants a picture of the crowd in front of the City Hall within 5 minutes, the Server will respond that it can deliver the picture with 65% probability. The customer may then either relax the time and accept the service; or, it may request that a taxi drives by the target at extra cost.

In the experiments we have also evaluated a very simple (lower bound) case where the vehicle does not know where it is going next. Then, when it receives the invitation from the Server, it either takes the picture if it is in the Shot Zone, or it drops the invitation (ie it does not memorize in case it lands on Shot Zone later).
3.3.2 Forensic Picture Service

The Pics-on-Wheels service can also be used to retrieve pictures taken spontaneously by cars as part of background ambient surveillance. Suppose vehicles routinely capture license plates, car makes and other surrounding images as they roam through the city streets. Then, a valuable service for forensic investigators is for the cloud to identify witness vehicles that take pictures in the Service Target from the past. For example, if the insurance company is investigating a vehicle collision, it will request pictures taken in a given intersection at the time of the accident.

How can a customer request and receive this service? The performance of Forensic Service Service depends on whether the Cloud Service Provider record historical trace of vehicles passing through service Time. If the Cloud Service Provider contains location/route update which showing vehicles passed Service, it contacts the vehicle for the requested picture. However, some vehicles may passed the Service Target without updated location in the Service Area. Similar to the Zone of service, we define an area called Zone of Witnesses, which refer to the area where vehicles may travel to after exiting the Service Area. At the beginning of the Service Time, the Zone of Witness has the size of Service Area, and Zone of Witness expands as the time goes. For any vehicle that performs a location update outside the Zone of Witness, it is instantly ruled out as a possible photo provider.

The procedure for requesting forensic photo is summarized in four steps: Step 1: The Cloud Service Provider receives the request from a customer computes Zone of Witness. Step 2: The Cloud Service Provider contact vehicles that have provided location update or route update in Zone of Witness. Step 3: For all vehicles in Zone of Witness, find the last location update before Service Time. Marked vehicles that are also in the Zone of Service on their last location update.
Step 4: Contact the Marked vehicles for Forensic pictures.

Here is an example of accident scenario. When accident happens, the Customer wants to see all pictures since the accident happen. From the difference current time - time of accident, the Cloud Service Provider calculates the Zone of Witnesses and also ZoS. When vehicle sends a location update, Cloud Service Provider checks if this location update is inside ZoW. In addition, if historical trace is available, it check if the last location update before the time of accident is inside Zone of Service. If so, the Cloud Service Provider sends out Job Notification to request for forensic picture service. If the vehicle actually traversed Service Target (ie street # in the query) in Service Time, it returns an OK committing to upload the images taken in Service Target. The Forensic Picture Service implementation is very similar to the Photo Shoot service. There are some differences in performance, however. The response time requirement is not as critical as in Photo Shoot, since the event has already happened. The success rate is proportional to number of vehicles that witness the accident, ie, that are in Service Area during Service Time. The Service Time constraint greatly restricts the number of successful matches with respect to the Photo Shoot service. To obtain a reasonable success rate one must rely on a large population as the one controlled by the Navigator Server (as opposed to the taxicab population controlled by the dispatcher).

In our study we will evaluate the performance of Forensic Photo Service as a function of vehicle density, query frequency, accident Service Time, for a given response time.

The careful reader will note that the Forensic Service is similar to Mobeyes [43]. However, Mobeyes is totally distributed. It required motorized agents to investigate the accident. In such cases, epidemic dissemination of vehicle metadata (ie, place, time, vehicle ID) was proposed to increase the probability of discovering witnesses. In our case, epidemic creation of proxy records is not required since
the probing of the vehicle cloud is centralized, via 3G.

### 3.4 Simulation Results

In this section, we analyze the performance of surveillance service with real world scenario. Previously, project cabspotting.org was launched in 2006. This project trace participated San Francisco taxis cabs as they travel through the bay area. Participating taxis are installed with GPS trackers, which periodically update its own location to the centralized location tracker center. Each location update is about 60 seconds.

![Figure 3.3: location of selected targeted location](image)

At year 2008, researchers collect 25 days of trace, with more than 300 running vehicles at any time. This dataset contains 500 taxi traces from 5/17/2008 to
6/10/2008. This data set currently is the vehicular ad hoc network (VANET) log with most mobile nodes. We assume that while these taxis report their location, there are also surveillance cameras which can be used to collect data. At each contact the Server can deputize the vehicle for Pics-on-Wheel service.

Figure 3.3 shows the targeted region. The Service Area is a 100 meter road section. We take the traces from 9:00 am to 9:00pm, and aggregate them into five 2-hour interval Service Time.

![Individual Visits / 2 hour](chart)

**Figure 3.4:** Service availability comparison between per minute location update(with cabspotting location update) and full path knowledge

In the first scenario, we assume that the vehicles are able to take 1 picture once Job Notification is received. We show the result based on 2 different update scenarios, shown in Figure 3.4. The bottom line of Figure 3.5 is the case where Cloud Service Provider only send Job Notification when the vehicle is inside Service Target. During location update, the vehicle must either immediately commit to take the photo, or decline. Because the Cloud Service Provider's knowledge of
Figure 3.5: Cumulative distribution between a Service Request arrives Cloud Service Provider and the Service is completed

the path, it can notify vehicle only if the location update happens to be in the Shot Zone. On the other hand, the top lines show the number of available picture taken if the vehicle provides complete route information, or the Cloud Service Provider contact all vehicles through the Zone of Service. Note that these two scenarios provide identical results as they all consider every possible vehicles.

The result shows that the number of photo shots taken by the path-unaware vehicles, (ie cabspotting time) is smaller than that of the path-aware ones or the one update through Zone of Service. In fact, it is about 3 to 6 times smaller depending on the traffic congestion for both case A and B. This is because the path aware vehicles can check in every 10 to 20s for a Service Target match (as it takes 10 to 20s on average to traverse a Service Area in light or heavy traffic conditions), while the less path-aware vehicles have such opportunity only every 60 seconds, at cabspotting instants. These results show that path awareness is
important for efficient Pics-on-Wheel service delivery. In fact, the longer the update interval, the more critical path-awareness becomes.

Figure 3.5 shows the CDF of service response time by real traffic traces, assuming the Customer request for the latest picture possible of one location. In other word, the Service time starts from current time to Infinity. In such cases, because there is no end time for Service Time, Zone of Service cannot be bounded. Thus, the number of candidate vehicle is only bounded by the location update. Figure 3.5 shows the delay difference between services only using location update, and services based on path knowledge. Without path knowledge, more than 50% of the requests take longer than 4 minutes to complete. However, with path knowledge, 80% of the requests are completed within 2 minutes. This result shows that even with just 500 taxicabs, POW service is able to provide good service coverage to a road section. The results confirm the importance of path awareness. Of course, path awareness will be the rule when the Service is offered by the Navigator Server or Cloud Service Provider.

Figure 3.6 and 3.7 show the simulation results for Forensic Picture Service. The service is limited not only by the service area, but also by the service time window when accident occurs. Each vehicle sends now an update once every 210 minutes, less than before because the number of vehicles has increased and overhead must be kept at a reasonable level. We ran the simulation using the same target as in Figure 3.3 requesting a picture that was taken within the last 5 minutes (ie ToS = 5 min), with a 30-second time window. In order to increase traffic density, we take taxi traces collected from different dates and fold them into the same day. For example, by aggregating 10 2-hr traces, we are able to emulate a scenario with more hardware providers than the number of vehicles provided by the data set.

Figure 3.6 shows the scenario when an accident happen and request for a Forensic picture from the past is sent. Different lines represent the frequency
Figure 3.6: The average number of picture collected with different number of vehicles for a targeted 30-second window.

for a Hardware Provider to report to the Cloud Service Provider. Suppose the Service Customer requests a deadline of 5 minutes. If the vehicle location update frequency is below 5 minute, then all vehicles collected the picture would submit to the Cloud Service Provider. Therefore, the number of picture collected within 2 minute interval and 4 minute is the same. After the location update, the interval exceeds the picture collection deadline, the number of picture collected decreases as the update interval increases. In addition, by increasing the number of vehicles, the number of pictures increases proportionally. This is important because the vehicle updates their location voluntarily, and Cloud Service Providers can not affect how fast a vehicle update their location. By having more participating hardware providers, the Cloud Service Providers can still collect enough pictures.

Finally, figure 3.7 shows another scenario where Service Customer request for the fastest respond picture from the given window. If the picture is not received
Figure 3.7: Success rate to finish the service within 5 minutes.

within 5 minutes, the Service Request is considered unfinished. This figure shows the success rate to demonstrate the number of vehicles to be contacted in order to complete service. For example, with only 500 vehicles, the success rate is only 0.9 %. If the vehicle number increase to 5000 vehicles, the success rate climbs to 78%.
3.5 Summary

Vehicles and Vehicle Clouds provide a perfect platform for on demand location services. This chapter has described a vehicle cloud service that exploits vehicle predicted routes to provide photo shots of target locations on demand. Key to performance is the ability to identify the candidate vehicles that take the shots. Based on given location update interval, we plot the number of vehicles that Cloud Service Provider should contact in order to achieve target success rate. The study shows that even a modest number of vehicles yields latencies in the order of 5 minutes in well frequented area. With further monetary incentives to drivers, detours can be triggered to less popular areas with good results.
CHAPTER 4

Realistic Pics-on-Wheel Service

4.1 Overview

The concept of using a vehicular cloud to provide as surveillance service, Pics-on-Wheels, is discussed by our study in the previous chapter. The service is an on-demand photo shooting service when vehicles report its locations periodically at the time scale of minutes. The photo service request is distributed to the service vehicles before they enter the service area. From the study we notice several things. First, without knowing the route of a vehicle, the service either would have to notify lots of redundant vehicles driving near the target area. The longer the location update is, the number of vehicle involved increases squarely. Secondly, although increase the number of participating vehicles reduce the service delay; the latency improves logarithmically while the cost increases linearly.

In this chapter, we improve Pics-on-Wheels with the following methods. Vehicles report their position as fast as the hardware provides. While this increase the cost of location bandwidth consumption, the resource consumed is acceptable for each participating vehicle. To reduce the cost of recruiting participating vehicles, we ask the participating vehicles to detour and take pictures proactively. We show that by detouring for 100 meters, the system performance increases significantly.

The rest of the chapter is organized as followed. Section 4.2 discusses the open mobile cloud requirement. Section 4.3 discusses the design of Pic-on-Wheel service, which is implemented in Section 4.4. Section 4.5 presents the use case
analysis assuming Pics-on-Wheels service is installed among San Francisco Taxis. Section 4.7 discusses the related work. Summaries and future work are in Section 4.8.

4.2 Open Mobile Cloud Requirement

This section discusses the basic open mobile cloud ingredients and performance requirements to provide location services. First, the Hardware Provider vehicles are equipped with GPS for periodic location updates to the Cloud server. Second, the vehicle hardware provides service to Cloud Service Provider. Third, infrastructure network support must be available to update position, receive job requests and delivery data to the Cloud Service Provider.

The Cloud Service Providers act as a broker between the vehicle and the Ser-
service Customer. It manages participating vehicles in the mobile cloud and keeps track of the time and location updates for each vehicle. When one Service Customer sends a job request, the Cloud Service Provider selects the proper vehicles to perform the job. The Service Customers are the actual consumers that use the service. Each request contains at least a triple of [Location, Time, ServiceType].

By decoupling the Service Providers and the vehicles, direct connections are not required between vehicles. However, a vehicle may provide the service to help its neighboring node connect to the Cloud Service Provider. Therefore, neighboring nodes create different depth of connectivity in the hardware-providing tree. Since location information is centralized, job requests can be delivered to the proper location more quickly and efficiently, without relying on the vehicles to perform multiple levels of broadcasting.

In order to provide location based services, the vehicles must be able to locate themselves, usually via a GPS. Moreover, Cloud Service Providers should take into account the inaccuracy of GPS. Current commercial GPS in the navigation
systems does not provide good enough location accuracy. The location error for navigational GPS ranges from 10 to 15 meters. There are two reasons behind it. First, the speed of the driving car is too fast, and improved GPS precision does not increase navigation accuracy. Secondly, the underlying map used for driving direction is not accurate enough as well.

For surveillance services, GPS with Wide Area Augmentation System (WAAS) [44] is suggested to reduce its geographic error to less than 3 meter. Commercial WAAS-GPS can be used as external devices for laptop and tablets. In general, the update frequency of current commercial GPS ranges from one to five Hz. For a vehicle traveling 65 mph, a 5Hz GPS has an error margin of less than 6 meter. To sum up, a WAAS supported GPS with 5Hz location update frequency can guarantee the geographical error to be less than 10 meters. Since not all vehicles have high quality GPS, the error range should be reported to the Cloud Service Provider to provide reliable service.

The vehicles in Mobile cloud advertise their positions. In general, a location update from 10 seconds, 1 minute, and 5 minutes ago implies the hardware is 200 meter, 1 km, or several km away from the reported location. For real time applications, vehicles should try to update their locations to the system through 3G connection as fast as the GPS provides (i.e. 5 times per second). For a location update packet, latitude and longitude stored with 4 bytes of data provides 2cm precision. UNIX timestamp uses 4 bytes to provide precision up to one second, and a maximum of 5 bytes is needed to provide enough time precision. Overall, each location update requires only 13 bytes of data payload.

Under current TCP/IP standards, assuming each update is sent to the server via UDP packets, the bandwidth consumption for a location update is (IP header + UDP header + Data Payload) * Update Frequency = (20+8+13)*8*5 = 1640 bit/s. Suppose a vehicle drives 4 hours a day, and 30 days a month, the amount of data for location update is 84.46 MiB/month. Standard US rates for 1GiB of
data is around $10, which translates to less than 1 dollar/month for the cost of location updates. This shows that if the vehicle wish to provide the best location they could, the bandwidth consumption is reasonable.

In practice, location updates are much less frequent because not all jobs require instant service. For future job request, as long as the job is delivered to candidate vehicles before the service time, the update frequency does not matter. However, a reasonable update period still reduces the control overhead. Figure 4.1 shows an example of a job request targeting location L. If every vehicle updates once every 5 seconds, The Service Cloud Provider only needs to contact vehicles which can travel to L in less than 5 seconds. Finally, the cost of updating GPS should not be charged to Vehicles to increase the incentive to participate in the Mobile Cloud Service.

When the vehicles are eligible to provide service, there could be two approaches automated service and interactive service. For automated services, once the vehicle opts-in to become a member of the mobile cloud, it provides service as a background process of the hardware. In other words, the hardware leasing process is invisible to the vehicle, and the cloud service does not interact with the driver while finishing request from the customer.

For interactive services, the drivers actively participate in the service and react to job requests. Interactive service happens when the job requires human interaction such as pressing a hardware button or detour driving. For example, suppose the Cloud Service Provider also provides the navigation information for the driver, it can inform the driver to detour from its current route to complete a near-by job request. This is effective when the service target location has low vehicle traffic volume.

In Section 4.5 we show that when the drivers are willing to detour for a target 100 meters away from the current route, the performance greatly improves. Since interactive services require the drivers action, the Cloud Service Provider should
provide some form of incentive that profits the driver for participating in the system.

4.3 PICS-ON-WHEEL SYSTEM DESCRIPTION

The Pics-on-Wheels service architecture is illustrated in Figure 4.3. The vehicle takes the picture on demand. It is connected to the Navigation Server via 3G for location update (from car to Server); push notification service (from Server to car) and; picture/video upload (from car to server).

Fig 4.3a shows the software architecture in Cloud Service Provider. The vehicle constantly updates its location to the server via UDP packets. When Pics-on-Wheels server wishes to contact vehicles in a certain area, ideally it would perform a geo-cast to all vehicles close to the service area (i.e. photo shoot area). However, geo-cast is not a standardized 3G service. Moreover, today’s mobile endpoints are hidden behind the firewall and NATs. Connections can not be initiated by the server and must be initiated by the vehicles. To get around this problem, we adopt a technique similar to the email push service. The vehicle starts a TCP connection to the Job Notification Server. The Job Notification Server then uses this channel to push job request to the vehicle. Once the data is collected, the vehicle can upload the picture via 3G or Wi-Fi.

Fig 4.3b shows the vehicle software framework. The GPS, camera, and network devices are required hardware. The software is represented by boxes. The network status monitor constantly checks the connectivity between the mobile node and infrastructure. If it detects handoff, IP change or new device availability, it notifies Task Notification service and GPS updater to connect (or reconnect) to the Pics-on-Wheel server.

GPS Updater constantly sends out UDP packets to update the position of the vehicle (position information is stored server side in the location database),
(a) POW Cloud Service Provider

(b) POW Hardware Provider

Figure 4.3: Pics-on-Wheel service architecture
while the Task Notification service maintains a connected TCP channel to the Job Notification Server. This channel is used for Pics-on-Wheel server to push jobs into the car, as described in the previous segment.

When a Service Customer uses the Pics-on-Wheel service, a request is sent with location and time requirements. The job management server will check the location database to find the proper candidates and assign the jobs. When the on-board service handler receives the job, it checks GPS for time and position. Then, it takes the picture if the vehicle is in the correct place and within deadline, and uploads the data. For the automated service, the entire process is done automatically. For the interactive version, the service handler informs the driver of job request and reward. The driver must decide whether to detour and collect the reward.

### 4.4 Implementating Pics-on-Wheel Service

Figure 4.4: Pics on wheel hardware setup and photo example.
Figure 4.5: Pics-on-Wheel Latency Breakdown

We setup a prototype of Pics-on-Wheel using commercial off-the-shelf equipment. The Mobile Cloud Service is implemented and installed on a desktop with Intel Due-Core Pentium G645 processor and 4GB of ram with Ubuntu 12.04 LTS. The server opens a UDP port for location update, a TCP port for both push notification and data upload.

The Vehicle is set up on a laptop with an Intel Core 2 Duo CPU, 2GB of RAM and 120GB hard drive, as shown in Figure === ===. The laptop is connected to a GPS with SiRFstarIII chipset, which includes 20-channel receiver and WAAS support. The laptop is also equipped with a webcam that supports 8 mega pixels photo and 720 pixels video. The laptop connects to the Vehicle through AT&T
3G network for location update and job request/data responses.

Figure 4.4 shows the example of a hardware provider vehicle and the sample picture taken. The webcam is placed on top of the rear view mirror, as shown in the top red circle. The bottom circle is the USB GPS connected to the laptop at the bottom of the picture. Example pictures were taken while driving and shown on the right hand side of Figure 4.4. The testing shows that commercial webcams are able to provide photo surveillance with clear quality.

Next, we test the time need for the Pics-on-Wheel service to perform a service. First, a job request is initiated from the Cloud Service Provider, which request for current picture at the location of the vehicle. Once the vehicle receives the request, it immediately returns an ack to the Cloud Service Provider. Then, it verifies its own GPS and timestamp to take a picture, and sends another ack after the picture is taken, followed by uploading the picture to the Cloud Service Provider.

Figure 4.5 shows the latency break down in the experiment. The time between a job request and data received varies between three to four seconds. However, the major delay happens on the picture taking process, which generally takes up to two seconds to complete. In addition, the round trip delay for 3G connection ranges between 200 to 400 ms, and the transmission delay to upload a 150 KiB picture is about 500 to 1000ms. The result shows that network delay for reaching the vehicle is at the scale of few hundred milliseconds. Secondly, to improve the round-trip-delay of the Pics-on-Wheel service, a customized photography application is needed. Finally, if the target location for a job request needs to be really accurate, it would be better to use multiple pictures or video to increase the service reliability.
4.5 Use Case Analysis

In this section, we evaluate the Pics-on-Wheel service based on real world traces and scenarios. The San Francisco Taxi cab trace [4] contains 536 participant taxis in an experiment run from May 17 2008 to June 10 2008. We simulate the performance of our service by assuming these taxis all participate in Pics-on-Wheel and provide photo service. During simulation, customers contact the Cloud Service Provider at random times, and request the photo of the same location. The Cloud Service Provider dispatches the job request to all candidate vehicles. Any vehicle driving pass the target takes a picture and uploads it to the Server.

![Figure 4.6: Location of traffic samples](image)

We repeat the process for 5 locations in San Francisco as shown in Figure 4.6.
Figure 4.7: Traffic Volume for undetoured traffic

<table>
<thead>
<tr>
<th></th>
<th>Visits</th>
<th>Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>27.97</td>
<td>235.5 s</td>
</tr>
<tr>
<td>B</td>
<td>23.63</td>
<td>206.5 s</td>
</tr>
<tr>
<td>C</td>
<td>1.52</td>
<td>13.7 s</td>
</tr>
<tr>
<td>D</td>
<td>4.27</td>
<td>39.0 s</td>
</tr>
<tr>
<td>E</td>
<td>77.16</td>
<td>273.6 s</td>
</tr>
</tbody>
</table>

Figure 4.8: Service delay provided by POW service in 5 different locations

Figure 4.8: Service delay provided by POW service in 5 different locations
Each location is a road segment ranges from 90-120 meters. These locations are selected to show the performance difference between road segments with different levels of vehicle traffic volume. Locations A to D are local street segments with different traffic volume, and Location E is a Freeway segment near San Francisco International airport.

Figure 4.7 shows the average number of vehicles per hour passing through each segment during the observation interval. The Visits column shows the number of taxis entering the road segment and taking photo, while the Covered column is the time in seconds when at least one taxi is on the road segment.

Figure 4.6 shows the cumulative distribution function (CDF) of the photo latency of a Mobile Cloud Service. Location A and B are areas with more traffics, while location D and E are road with a lot less traffic. Location E is, as mentioned earlier, a freeway section connecting SF downtown and San Francisco International Airport. The result shows that 80% of the photograph service can be completed within 300 seconds in A and B locations.

<table>
<thead>
<tr>
<th></th>
<th>Visits</th>
<th>Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>27.97</td>
<td>235.5 s</td>
</tr>
<tr>
<td>B</td>
<td>23.63</td>
<td>206.5 s</td>
</tr>
<tr>
<td>C</td>
<td>1.52</td>
<td>13.7 s</td>
</tr>
<tr>
<td>D</td>
<td>4.27</td>
<td>39.0 s</td>
</tr>
<tr>
<td>E</td>
<td>77.16</td>
<td>273.6 s</td>
</tr>
</tbody>
</table>

Figure 4.9: Traffic Volume for taxi willing to detour for a target 100 meter away

Next, we assume the taxi drivers are willing to detour to finish the job request for an extra reward. Suppose the drivers are willing to detour for maximum 100 meters, which in general costs the driver less than 1 minute extra time. Since
more vehicle now are able to reach Service Target, the traffic volume increases to two to three times, as shown in Figure 4.9. The simulation result shows that the number of visits is increased by two to three times. Figure 4.10 and Figure 4.11 shows the CDF of the photo latency of with 100 meter detour. The number of service completions within 300 seconds increases from 12% to 17% for location C and 28% to 46% for location D. As for location A and B, the service completions within 300 seconds increase from 80% to 90%.

4.6 Service Rate Modeling

In this section we try to model the service latency by observing customers which see no vehicles at the target location when doing job request. First, we assume that the vehicle arrival is a Poisson distribution, and calculate the arrival rate using Table 4.7 and Table 4.9. Since we know the total number of vehicles arrives every
Figure 4.11: On demand service latency with maximum 200m detour

hour, the estimate service delay distribution would be an exponential distribution. We compare the actual Service Delay with the modeled Poisson distribution at location A and E in Figure 4.12. Although the trend is similar, we found that using Poisson distribution to model service arrival under estimates the actual delay. After observing the raw trace, we discovered from the taxi traces is that taxis tend to arrive in clusters as opposed to completely randomly as in Poisson.

We select random location from the simulation map and collect service delay CDF for random arrival Service Request. Using least squares regression, the CDF with exponential distribution. For each location, the rate lambda for the fitted exponential ranges 71% 80% of the actual incoming vehicle rate. Figure 4.13 and 4.14 compare the service delay distribution and adjusted Poisson approximation using simulation location A through E. We found that on location A and E, it provides much better result comparing with non-adjusted Poisson approximation.

To sum up, although the arriving vehicles can not be modeled as Poisson
arrival, we found that the service delay can still be approximated as an exponential distribution through numerical fitting. Based on our observation, a 75% of rate adjustment provides good approximation.

![Figure 4.12: Actual service delay and approximation with Poisson distribution](image)

### 4.7 Related Work

In [45], Camp and Knightly shows existing rate adaption algorithm does not work well for the outdoor environments, and implement a coherence time training module to select the best bit rate. BRAVE [46], a design and implementation of rate adaptation algorithm, targets vehicle-to-vehicle communication environment. Since the speed of vehicles creates different channel dynamics than passengers, these must be taken into account for vehicle to vehicle (V2V) communications. In [47], the authors proposed to use GPS to select static access points and optimal bit rates for mobile devices to save energy. [48] shows the realistic limitation of using Wi-Fi infrastructure in VANET.
Figure 4.13: Actual service delay and approximation with rate adjusted Poisson distribution on location A and E

Figure 4.14: Actual service delay and approximation with rate adjusted Poisson distribution on location B,C,D
4.8 Summary

Vehicles and more generally Vehicle Clouds provide a perfect platform for on demand location services. This chapter has described a vehicle cloud service that exploits monitored vehicle routes to provide photo shots of target locations on demand. Key to performance is the ability to identify the candidate vehicles that take the shots. The study shows that even a modest number of vehicles yields latencies in the order of 5 minutes in well frequented area. With further monetary incentives to drivers, detours can be triggered to less popular areas to provide better results.

So far, the service we proposed only uses cellular communication as control messages. In the future, we will extend our service to use vehicle to infrastructure and vehicle-to-vehicle communications. If vehicles take into account the GPS location of surrounding access points provided by the Cloud Service Provider, better performance handoff can be achieved.
CHAPTER 5

Conclusion

The mobile cloud computing is being defined as the combination of cloud computing and mobile devices. The mobile devices are considered as clients and sensors, which provides interface for human interaction and/or collecting data. Today smartphones and tablets are used for accessing email, posting pictures on social networks, sharing videos, and crowdsourcing traffic congestion. The idea for offloading computing from mobile devices to the Internet cloud overcomes obstacles related to performance and reliability, allowing the usage of new services and applications.

Although mobile devices provide end-users a new way to access the Internet cloud, the system architecture of the client - data center model is not new. From the systems perspective, there is not much difference between accessing the Internet cloud through wired or wireless networks. The two major differences between mobile and non-mobile clients are (1) Hand-held devices are more convenient data collectors (e.g., camera, motion) and are generally equipped with Global Positioning System (GPS). (2) Wired network provides cheaper and faster network access for desktops. In both scenarios, the Internet cloud is the powerful time-sharing system that provides massive computing power and storage, while end-users share the same hardware inside data center across the Internet. In mobile cloud systems, which have the similar time-sharing idea, all devices are distributed in the mobile network and can share resources and content.

A mobile cloud can be defined as a group of mobile devices that collaborate
with each other to provide powerful services. Mobile clouds can be situated at the edge of networks (e.g., edge clouds, cloudlets, or groups of mobile smartphones or vehicles). All data collection, process, and storage schemes are performed on mobile devices. Mobile clouds are better than the Internet clouds in two major areas – bandwidth consumption and localization.

In this thesis, we demonstrate the benefit of mobile resource sharing in two application. First, we build the PepNet - a virtual machine based experiment platform. If vehicles provide storage and computing power to host virtual machines, academics can exploits the participate vehicle and perform wireless experiments. Moreover, PepNet has made protocol performance evaluation easier based on its side by side comparison.

Another scenario we investigate is the surveillance application - Pics-on-Wheel. Suppose that the police department is looking for a specific target and requests mobile devices to submit their videos from the last hour. It would be more efficient, however, for each device to search for the target picture and then upload only the search result with processed information. When Pics-on-Wheel customer request for service at a low traffic density Service Target, we show that by requesting vehicle to detour for few hundred meters increase performance by 2 to 4 times.

Visualization on mobile device is an ongoing industry. As the computing power, storage, and network ability of in-vehicle increases, it would be nice to provide the spared resource for those in needed. By grouping vehicles to form a vehicular cloud, many additional application can be deployed.
References


[22] CMU wireless emulator “http://www.cs.cmu.edu/~emulator/”.


76

[31] Xen “http://www.xen.org/”.


[33] GPSD “http://gpsd.berlios.de/”.

[34] Wireshark “http://www.wireshark.org/”.


[37] RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing.

[38] RFC 3626: Optimized Link State Routing Protocol (OLSR).


