Vehicular Delay Tolerant Networking for Fleet Management Applications

by

Mark Stevens

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The objective of this thesis is the study and implementation of a Vehicular Delay Tolerant Network (VDTN) system for a fleet of vehicles, and the evaluation of its data carrying potential. The implementation relies on commodity hardware and communication using "WiFi" (IEEE 802.11) transceivers. We also detail the steps necessary for the accurate simulation of realistic, daily routines, of vehicular fleets serving an urban road network. We use as our example a fleet of service vehicles operating in the city of Lethbridge, Alberta. We analyze the dynamics of encounters among fleet vehicles throughout a typical working day, and introduce a Markovian model capturing the encounter distance dynamics. We can then translate the encounter distance dynamics to, corresponding, communication throughput dynamics. We perform data collection of IEEE 802.11 point-to-point throughput vs. distance measurements, which, in conjunction with the Markovian model, allows to derive the expected data carrying volume introduced by the VDTN. The results demonstrate that the data carrying capacity of the VDTN exceeds what is needed by typical vehicle monitoring applications. The surplus capacity can be used for delivering value-added services, such as data collection from external wireless sensor networks.

Acknowledgements

First I would like to thank my advisor, Dr. Ioanis Nikolaidis, of the Department of Computing Science. He provided continuous support during my stay as a graduate student as well as extensive contributions towards the completion of my thesis. His eye for aesthetics was immensely helpful and I am quite sure that this thesis would not have been completed if it were not for his countless hours spent improving the thesis text with me. Moreover, his natural inclination towards wanting to create and test real systems was a perfect fit for me as a student. Lastly, on a personal note I have always enjoyed his sense of humor and various unrelated topics of discussion we have had over the years.

I would also like to thank my family and friends for their continued support during my academic journey. In particular my parents have always provided a foundation for which to stand upon, and guidance throughout the years.

Special thanks to my loving spouse, Julie, for her unending patience and inspirational words in times of need. She continues to amaze by accepting me for who I am, making my life easier, and balancing out my deficiencies.

I should also thank several colleagues who, at the time of their assistance, were completing their undergraduate work at the University of Alberta. Omar Almokdad helped considerably with respect to setting up the initial codebase for the data acquisition portion of the proof of concept prototypes that I worked on. David Skrundz spent a semester testing and reducing the overall bootup time of the Linux Kernel that we were using on the Raspberry Pis. Lastly, I would like to thank Brendan Gluth who worked tirelessly exploring the possibilities of incorprating netlink socket programming into the codebase as a way to clean up some of the logic involved in managing

IBSS networks.

To the members of my examining committee, Janelle Harms and Ehab Elmallah, for their time in reading my thesis and offering thoughtful comments and suggestions for me to consider, I thank you. A good portion of my graduate course work was completed under their instruction, which means they helped shape some of my fundamental understanding of the research being conducted in the area of computer networks, with emphasis on wireless communication.

I would be remiss if I did not extend my gratitude once again to my mom, dad, and brother for their help in conducting my field experiments. Not only did they graciously lend some of their equipment (ATVs, generators, etc), but they, along with my spouse, spent countless hours helping me actually carry out these experiments on location. My dad in particular also helped with various preparations. It was a family affair and I throughly enjoyed it.

This work was supported through funding from NSERC's Engage and Discovery Grants programs. The interaction with the Engage grant SME partner, Latium Technologies, helped inspire the direction of this thesis.

Contents

List of Figures

List of Tables

Abbreviations and Notation

Abbreviations

AP Access Point

ARP Address Resolution Protocol

BSS Basic Service Set

C-V2X Cellular V2X

CAN Controller Area Network

CM Connection Manager

DAD Duplicate Address Detection

DFS Dynamic Frequency Selection

DHCP Dynamic Host Configuration Protocol

DSRC Dedicated Short Range Communications

DTMC Discrete Time Markov Chain

DTN Delay-Tolerant Network

EDCF Enhanced Distributed Coordination Function

EID Endpoint Identifier

EIRP Effective Isotropic Radiated Power

ESSID Extended Service Set Identifier

FCC Federal Communications Commission

GI Guard Interval

GUI Graphical User Interface

HT High Throughput

IBSS Independent Basic Service Set (802.11 “ad hoc” mode)

IEEE Institute of Electrical and Electronics Engineers

IMAP Internet Message Access Protocol

IR Initiate Radiation

ISM Industrial, Science, and Medicine radio band

ITS Intelligent Transportation Systems

LTE Long-Term Evolution

MAC Medium Access Control

MCS Modulation Coding Scheme

MIME Multipurpose Internet Mail Extensions

MTA Mail Transfer Agent

OBD-II On-Board Diagnostics II

OBSS Outside of Basic Service Set

OBU On-Board Unit

OppNet Opportunistic Network

P2P Peer-to-Peer

PHY Physical Layer

PKI Public Key Infrastructure

POI Point of Interest

RNG Random Number Generator

RPi Raspberry Pi

RSU Roadside Unit

RTC Real Time Clock

S/MIME Secure Multipurpose Internet Mail Extensions

SDR Software Defined Radio

SIM Subscriber Identity Module

SME Small and Medium sized Enterprise

SMTP Simple Mail Transfer Protocol

SNR Signal-to-Noise Ratio

SS Number of Spatial Streams

SSID Service Set Identifier

TCP Transport Control Protocol

TCPCL TCP Convergence Layer

UDG Unit Disk Graph

UDP User Datagram Protocol

UNII Unlicensed National Information Infrastructure

V2I Vehicle to Infrastructure

V2V Vehicle to Vehicle

V2X Vehicle to Everything

VANET Vehicular Ad Hoc Network

VDTN Vehicular Delay-Tolerant Network

VHT Very High Throughput

WAVE Wireless Access for Vehicular Environment

WEP Wired Equivalent Privacy

WLAN Wirelss Local Area Network

WPA2 WiFi Protected Access 2

WSN Wireless Sensor Network

Notation

π_i Steady state probability for state i, corresponding to the probability the distance between two nodes is within the i–th interval (state) of the DTMC.

b_i Estimated bit rate between two vehicles in a given interval state of our DTMC

B_{end} The end of the working day during SUMO simulation runs.

B_{start} The beginning of the working day during SUMO simulation runs.

C_{VDTN} Approximated overall data carrying capacity due to encounters between vehicles of our SUMO simulation runs.

D_{max} Distance threshold defining the beginning (once less than that) and end (once exceeded) of an encounter between a pair of nodes. It is equivalent to the distance, up to which, communication between two nodes is considered possible.

$Data\_n$ Individual sensor data file obtained through the Data Acquisition portion of our proof-of-concept implementation.

N_c Current number of traffic vehicles during SUMO simulation runs.

N_d Delta value used as threshold for establishing lower and upper bounds of traffic vehicles during SUMO simulation runs.

N_t Desired number of traffic vehicles during SUMO simulation runs.

p_{ij} Conditional probability describing whether, if the distance of two nodes is within the interval i, it will be in interval j in the next time step. Here, *interval* relates to states of distances between vehicles. Represents transition probabilities between states of the DTMC.

T_{rt} Cutoff time with which *fleet* vehicles use to determine when they should start returning to the depot during SUMO simulation runs.

T_r A fixed amount of time before the end of the *working day* used to determine T_{rt}. It is a fixed value determined heuristically.

T_{sp} The length of time in seconds past the official start of the work day with which *fleet* vehicles are permitted to leave the depot during SUMO simulation runs.

T_{st} Length of time *fleet* vehicles remain stationary during customer service stops during SUMO simulation runs.

Chapter 1

Introduction

The present thesis was motivated by an attempt to *design and implement* a vehicular data network that captures the needs of vehicular fleet management applications. The followed principle was to attain reduced cost of operations, in particular in terms of equipment and networking services. For those reasons, emphasis was placed on using commodity platforms, existing wireless networking protocols, and any freely pre-existing infrastructure. While a substantial amount of research has appeared in the scientific literature pertaining to vehicular/mobile data networking, it is rarely accompanied by open source implementations. Additionally, performance results that have appeared in previous studies have not been revisited to make use of recent wireless protocol improvements. Consequently, one has to synthesize ideas at the intersection of Delay Tolerant Network (DTN) systems, vehicular networking, and Wireless Sensor Networks (WSNs) and to synthesize, from scratch, an implementation. An additional challenge addressed here is how to evaluate an example implementation.

1.1 Vehicular Fleet Data Services

The motivation for the thesis work was a case study for a fleet of vehicles that drive over urban or rural roads, and, in extreme cases, in scenarios involving off-road or unpaved roads travelled, e.g., in logging operations. On-board equipment collects a lot of information from the various sensing systems of the vehicle from the car's on-board engine computers via e.g., the On-Board Diagnostics II (OBD-II) protocol. The full resolution data collected can be used

for off-line analysis for tasks such as the predictive maintenance of the vehicles, or the performance assessment of the drivers. Several vehicular fleet examples exist: delivery services, distribution centers, utilities repair staff, sales representatives, telecommunication installation technicians, or industrial companies whose needs require staff to drive particular heavy duty vehicles. In all scenarios, it may be desirable to track the movement of the vehicles and monitor the driving habits of employees. Services and decision making can then be based on the collected data, e.g., policies for future driver training, prevention and defense against traffic violations and accidents, etc. Note that by restricting the focus to fleet management applications, we are automatically excluding data communication related to activities such as entertainment or any other non-business tasks.

While highly dependent on the exact application, one can roughly separate the data pertaining to fleet management into two broad categories. One category contains data that are desired to be communicated in real-time, because their utility crucially depends on prompt delivery. An example is the dynamic dispatch of a service vehicle that is already underway to new locations, as unpredictable demands arise. The second category are data that are ingested by systems for the purposes of decision making, e.g., vehicle maintenance scheduling, over longer periods of time. That is, the second category of data can be delivered "delayed" without any noticeable impact.

Observation 1: A rough separation of the data handled for vehicular fleet operations is one between real-time and non-real-time data.

The focus of this thesis is the second, non-real-time, or "delayed" category. Most vehicle fleets today employ various forms of, usually mobile, user terminal equipment. The terminal equipment is used by the vehicle operator, e.g., to enter delivery details, or to send and receive notifications to the head office. The communication facilitated through the terminal equipment is usually "real-time" in appearance because it involves short message exchanges meant for user interaction, but we note in the next section that it does not deserve the real-time characterization. On the other hand, non-real-time data

can, and may have to, involve larger data transfers and are not involved in interactive applications. We therefore correspond the real-time communication to short message interaction exchanges and the non-real-time to large bulk data transfers. Note that by ignoring the case of real-time transfers of large amounts of data, we are eliminating from consideration "exotic" options, such as, fleets of vehicles equipped with cameras, streaming live in real-time their point-of-view.

1.2 Vehicular Communication Technologies

Short message exchanges are already supported by cellular data services and remain one of the most successful applications in telecommunications history. However, for large data transfers over cellular there still exist three shortcomings: (a) transfers may not be able to complete promptly due to spotty coverage or high cellular network load, and, more importantly, (b) often the fee structure penalizes (via caps and data overage rates) the large volume users, compounded by the fact that, (c), each user is a separate identifiable subscriber subjected to fees. Put simply, a fleet of N vehicles needs at last N distinct Subscriber Identity Module (SIM) cards and connection fees. Even when (c) has to be absorbed because the fleet operators need a real-time capability as noted earlier, the combination of (a) and (b) raises the question of whether there are any inexpensive (ideally, free!) alternatives circumventing the data fees and with adequate data carrying capacities. Also (a) suggests that bulk transfers are often associated with an acceptable delay. As a very telling example from a different application, currently (in 2023), many smartphones offer the user the ability to backup their photos on the cloud, but users routinely enable this option only for when they are in proximity to WiFi Access Points (APs), to avoid data charges for their mobile phones.

Observation 2: There is often a, monetary, cost associated with the transfer of large volumes of data over cellular infrastructures.

The broader area of vehicular communication covers subjects making use of acronyms, such as, Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I)

and Vehicle to Everything (V2X), depending on the character of the two communicating endpoints. As will be detailed in the related work chapter, specialized protocols have been proposed, and are in the process of coalescing into standards. Yet, there is an overall lack of real products on the market, except for a very narrow set of options as we will see. The main reason is that there exists an entrenched option already, that of cellular communication, that addresses (even if not perfectly) a wide range of application needs – of course, at a price. Remarkably, cellular services are also considered inadequate for *true* real-time notification among vehicles in critical applications such as road safety scenarios. An implication is that the V2V literature emphasizes the need for special protocols precisely to address true real-time notification.

It is no surprise that recent iterations of cellular services (5G and beyond) are attempting to provide services that approximate "true" real-time messaging but their adaptation at scale is something that has not been demonstrated, even if we leave aside the cost aspect. While the subject of the thesis is non-real-time communication, a case can also be made that true (hard-deadline) real-time communication is in fact also an open area. The lesson derived from the entrenched cellular services is that a pre-existing standard and service model are difficult to replace, even if they are not exactly appropriate for new or extended needs. In this thesis we use this lesson to our advantage by using an entrenched technology to provide bulk data transfers – that of wireless "WiFi" services as we came to know the protocols of the IEEE 802.11 family. Notably, all in-production vehicles these days provide WiFi support, even if typically restricted to enter-/infotainment services of the passengers, i.e., as in-vehicle networking. Re-purposing an existing entrenched technology for fleet vehicle data transfers would incur no notable cost. A compelling advantage for using WiFi technologies is that they have evolved, while maintaining backward compatibility, to allow ever higher bit rates to be achieved.

Observation 3: WiFi (IEEE 802.11) communication is an inexpensive, already available, technology on vehicles produced in recent years.

From its inception, the WiFi family of standards, provided a few alternatives to organize the nodes involved in communication. The one that found ex-

tensive use was that of a “managed” network, i.e., one where a node plays the role of an AP, and client nodes connect to other and communicate with each other (and with the broader Internet) via the AP, thus forming a star logical topology. A degree of configuration is required to achieve this arrangement, i.e., the AP must emit specific beacons with station IDs, and the clients must be aware of which station ID they need/prefer to communicate through. Also authentication and encryption schemes are involved in the setup. Despite its resounding success, this model of communication is not the most flexible. An alternative, called the “ad hoc” mode, allows direct peer-to-peer communication but, as we will see at various points in the thesis, still requires a degree of a-priori configuration and, alarmingly, is not always usable at high bit rates available today. In other words, the IEEE 802.11 technologies are not the most conducive for spontaneous peer-to-peer communication. In adopting IEEE 802.11 we will inevitably have to come up with configuration solutions to allow the maximum performance be achieved out of the protocol.

1.3 Delay Tolerant Networking

Given the cost of cellular communication, especially in remote areas where satellite communications may be the only means to communicate, a cost-conscious fleet operator would rather delay the transmission of the data that could afford delaying, i.e., could opt for *delay tolerant* communication, and hence the attention to DTN technologies. We will implement DTN using IEEE 802.11 as the link layer technology. DTN research has been active for approximately two decades and one would expect that, by now, some form of open source implementation of a complete system would have been available to be used “off-the-shelf“. After extensive search, it became apparent that, while simulators for DTNs were available, as well as the occasional isolated protocol implementation (the “bundle” protocol discussed in the related work), actual complete implementations of applications and protocols running and demonstrating a complete DTN system were not available.

An often used property in DTNs is to exploit the mobility of nodes to deliver data. When mobile nodes, such as those on vehicles, come in proximity, they exchange data they collected or generated, anticipating that one of them

will get the chance to deliver the data to its eventual destination earlier than the others. The technique generalizes to many nodes simultaneously carrying the same piece of data, one of which delivers it first and it is that, first delivery, that defines the delay experienced for the particular data item. Such "infection" paradigms have been the basis of many data exchange protocols that differ in how aggressive their infection is to encountered nodes, and any metadata and state information they retain to optimize the infection process. A technicality is that delivered data items have to be expunged from all nodes carrying a copy of the data that has been delivered. DTNs' assumption about the underlying link layer technology are very flexible, allowing intermittent connectivity which characterizes highly mobile peer-to-peer scenarios.

Observation 4: DTNs pose no particular requirements on the underlying link layer protocols, thus it is possible to define DTNs over IEEE 802.11 link layer protocols.

1.4 Supporting Value Added Services

Given today's abundance of inexpensive storage, along with the information pertaining to the condition of the vehicle (engine readings, driver responses, etc.), the vehicles participating in the VDTN may also provide additional, value added, services by collecting data from other sources. For example, it is possible to exploit the fact that the vehicles may often be (but not at precisely known times) near various WSNs that are not connected to the Internet. When near a WSN, a vehicle can receive data that has been accumulated on the sensor nodes over a period of time, and then act as "data mule" to deliver such data to their destination. The concept of *data mules* has been one of the motivating examples for VDTNs, especially for application in rural areas away from high speed Internet. In effect, our VDTN becomes an alternative *mobile sink* for WSN data, collecting the data from disparate WSNs spread over large geographic areas. The main advantage again is that the WSNs do not need to be using expensive communication infrastructure (cellular or satellite).

Observation 5: VDTNs can, in addition to data collected relevant to the vehicle, also collect data from encountered sensor networks that lack Internet connectivity.

1.5 Evaluation Strategy

For the purposes of a thesis, building and deploying a VDTN system at scale to study it is cost prohibitive and administratively complex. Yet, we wish to retain as much as possible of the realism of an actual system. We present a compromise, where the evaluation is conducted by splitting it into three separate components. One component is the implementation of the application and protocol logic which was tested in contrived examples, e.g., single vehicle and single point of data collection. This step ensures that the implementation performed as expected in terms of the semantics of what was expected – as in the exchange and delivery of data. The second component is a simulation of a typical use case with an adequate degree of fidelity insofar the trajectories of the vehicles are concerned and the tasks they will perform over an entire day. This component is used to derive statistics about the encounters of the vehicles and, hence, the potential for data transfers among them. Finally, the third component is a field measurement study using commodity IEEE 802.11 transceivers. The third component provided throughput vs. distance measurements that were possible to plug into the results from the simulations to derive approximate values regarding the volume of data that could be handled by the simulated VDTN on a typical day.

Observation 6: Evaluating a VDTN implementation at scale is a challenging task, hence combinations of measurements and simulations can form the basis of an approximate performance study.

The case we consider for evaluation is one of a fleet of service vehicles in a medium sized Canadian city. As such it expresses a wide range of similar cases differing possibly only in terms of scale and covered area. In most scenarios we are aware of, the vehicles begin their working day, each to serve a

set of pre-determined destinations in a sector of the city, after which they return to their point(s) of departure (the depot(s)) at the end of working hours. Because of their starting and eventual point in their orbits being the same, the delay of any data produced and/or collected can be bounded because at the end of the day they can perform a final data transfer, e.g., via a locally installed IEEE 802.11 infrastructure, while parked at the depot. Hence, no data is delayed by the VDTN for more than the duration of their working hours. However, it is through their mobility that provide data transport capacity to absorb data *during* their trips. While our work is motivated by a primary (engine OBD-II data) and a secondary (WSN data mule) application, similarly behaving data sources could be supported as well, both of volume proportional to the trip duration (as the engine OBD-II data are) and encounter-based (as the WSN data mule).

Having bounded the delay, the most pertinent performance metric is one able to capture the data transfer capacity present that would not have been present without the VDTN logic. We call this the *effective VDTN capacity*, and it is the average throughput over a working day of pair-wise transfers among fleet vehicles. The effective VDTN capacity is possible because nodes encounter other nodes during their trips throughout the day. Note that in this thesis, we do not evaluate the various DTN data transfer policies of which many exist in the research literature, but restrict focus to the effective VDTN capacity as it is independent of exact data transfer policies.

1.6 Thesis Structure and Contributions

We organize the remainder of this thesis as follows: Chapter 2 presents related work, emphasizing the facets of VDTN design and approaches used for evaluating them. Also, the data link layer protocols often encountered in the context of VDTNs are discussed. In the same chapter we also introduce some methodological elements for the remainder of the thesis. Chapter 3 presents the architecture followed by the software implementation of the VDTN and in particular the communication and the “bundle” handling components and explains the rationale around them. Observations from a limited real-life use of the platform are also presented. Chapter 4 introduces the simulation

setup using the SUMO [13] simulator and describes the process of mapping the vehicle fleet on the map of the city of Lethbridge and the addition of any landmarks necessary. Chapter 5 presents the simulation results for the fleet vehicles expressed primarily as distance between vehicles of the VDTN. A Markov chain model is introduced to approximate the distance relation between vehicles as they move. Chapter 6 details the field experiments and the platform limitations encountered on the way to producing distance vs. throughput measurements. Chapter 7 reviews the results of the field experiments and introduces the bit rate measurement statistics into the Markov distance model of Chapter 5 to derive the effective VDTN capacity. The thesis is concluded in Chapter 8 with a review of extensions and challenges that can be pursued in future research.

In summary, the contributions of this thesis are:

- the implementation of a complete VDTN software system for fleet vehicles using commodity class hardware and WiFi as the communication technology,
- the description of a sequence of steps to accurately simulate the daily routine of a vehicular fleet in an urban road network, through an example for a fleet operating in the city of Lethbridge, Alberta,
- an analysis of the dynamics of contact times between fleet vehicle (or vehicle and points of interest) throughout a typical working day, and introduction of a Markovian model capturing the encounter distance dynamics,
- a data collection of modern IEEE 802.11 point-to-point throughput vs. distance measurements, which together with the Markovian model provides an estimate for the additional data capacity introduced by the VDTN.

The VDTN code is available at: `www.github.com/uofanetdev/fleetDTN`

Chapter 2

Related Work

One can claim that DTNs/VDTNs are the re-emergence of the historical concept of store-and-forward networks, with emphasis placed on the *storing* aspect, as messages can wait for a long period of time before they are transferred and delivered. As an example of early store-and-forward networks, there is the Unix-to-Unix (uucp) protocol[1] which formed the basis of early email and news forwarding systems. Modems would dial up to establish links between nodes, and those links were, intentionally, not permanent as they were charged by telecoms at voice line rates, compounded by the need to place long distance, even international, calls. Usually the email messages would be stored at a node for the next scheduled dial out and transfer to a remote node. Interestingly, the uucp protocol email addresses were an explicit sequence of node identifiers, i.e., an explicit source-routed path to instruct how to copy from system to system to reach a destination user's system. A quick comparison of uucp vs. DTNs can be thought as, in general, DTNs having less certainty regarding communication "encounters" and DTNs not making use of explicit source-routing paths as they depend on chance encounters among nodes.

The nature of vehicular communication presents a particularly challenging environment, so it is instructive to first consider what specific protocols have been proposed for such an environment. The reader should also be mindful that the area of VDTNs overlaps somewhat with that of Vehicular Ad Hoc Networks (VANET), with VANETs essentially struggling to, as much as possi-

[1]The genesis of uucp is attributed [31] to Mike Lesk of AT&T Bell Labs.

ble, have a completely connected network while all nodes are mobile. It is not that VANETs do not account for the occasional disconnection of the communication graph, but they usually make it their (obsessive) pursuit to ensure that it is rare and as much hidden from applications as technically possible. VDTNs on the other hand, accept disconnection to be the norm, and expect applications to be able to absorb such behavior and work with it.

2.1 The IEEE 1609/802.11p Family

The IEEE 802.11p extensions appeared as Amendment 6 in 2010 [24] and got incorporated into the IEEE 802.11 standard. The IEEE 802.11p transceivers are incompatible with the rest of ordinary WiFi protocols – most notable of which is the restriction of IEEE 802.11p channels to 10MHz width (as opposed to a minimum of 20MHz in WiFi) and its operation in the 5.85–5.925 GHz band, which is higher than the WiFi "5GHz" band. The special frequency band was set aside for Dedicated Short Range Communications (DSRC) for vehicular (V2V and V2I) communication. The assigned spectrum is wide enough to provide seven channels, one of which is a control channel, the remaining being called "service" channels. In keeping with the numbering of the channels successively at 5MHz increments, the channel numbers are 172, 174, 176, 178, 180, 182, and 184. The control channel is the "middle" channel, 178. The transmission power is one of the relative advantages of 11p reaching up to 33dBm and bit rates, depending on coding, ranging from 3Mbps up to 27Mbps. The protocol designers' intention was to achieve reliable communication up to a distance convenient for vehicular applications, but without an explicit consideration for large data transfers.

The IEEE 802.11p protocol contains both a physical (PHY) and a Medium Access Control (MAC) layer. Apart from striking PHY differences mentioned above, the MAC differences play a crucial role to providing an ad hoc operation for highly dynamic networks. It borrows a traffic prioritization similar to the IEEE 802.11e Enhanced Distributed Coordination Function (EDCF), according to which, there exist four Access Categories (AC0–AC3), where AC3 is the highest priority. The priorities are expressed with different arbitration slots, inclusive of the inter-frame spacing, and backoff. Clearly, the higher

the priority of the traffic, the shorter those intervals in order to seize access of the channel in advance of lower priorities. Different channels (control vs. service) have different arbitration slot and backoff parameters. One can also argue that the IEEE 802.11p MAC protocol comes closer to the promise of an ad hoc communication mode than the WiFi subfamily. This is expressed through the absence of a Basic Service Set (BSS) concept, and the transmissions take place in Outside of BSS (OBSS) manner. Hence, no association or authentication setup between peers is taking place at this layer.

Orchestrating the transmissions across the various channels is left outside the core IEEE 802.11p PHY/MAC specification and resides within a separate family of standards, often referred to as Wireless Access for Vehicular Environment (WAVE), formally known as the IEEE 1609 protocols. The IEEE 1609.4, IEEE 1609.3, IEEE 1609.2, and IEEE 1609.1 are, respectively, addressing multichannel operation of MAC sub-layer, networking services, security services, and upper layer functions. IEEE 1609.4 defines a time frame of 100 msec split into a 50msec control channel access and a 50msec sevice channel access. In other words, the IEEE 802.11p transceiver splits its time equally between control and (some) service channel. There are synchronization requirements that are beyond the scope of this introduction to ensure the timing of the cycles is accurate. From the standard it is evident that the intention of the control channel is not to be involved in massive data transfers but to the absolutely essential control and safety operations. Large data transfers seem to be a secondary priority, and in any event assumed that will be given lower priority in terms of access classes.

2.1.1 The Performance of IEEE 802.11p

Some of the earliest work of evaluating DSRC by Bai et al. [4] came up with a disappointing conclusion that there was no range that would be considered as solidly reliable communication range, that is, effectively zero loss probability, across a range of distances. All of the area around a transmitter was “gray area” insofar probability of reception of a transmission was concerned. A notable feature of this work was that it involved actual V2V communication, using a fleet of three vehicles on a highway. The mediocre packet delivery rate

of no more than 80%, and as low as 20%, dominated the measurements. Such performance is not a good match for situations when messages are supposed to be conveying safety-critical information. Later, Gao et al. [19] provide another V2V testbed result involving trucks on a dedicated track demonstrating significantly better packet delivery rate, which they attribute on the use of two antennas on either side of a truck. Nevertheless, they comment on the unevenness of the terrain and surroundings as having an impact on packet delivery ratio. In yet another testbed data collection, Teixeira et al. [51], seem closer to confirming the original observations by Bai et al. [4]. Namely, a testbed involving two vehicles produced an average transfer rate of 8Mbps using User Datagram Protocol (UDP) traffic at distances less than 100m – including very short distances. At longer distances the attainable transfer rate quickly dropped but what was evident across all distances was the very high transfer rate variability. Even at 100m the 8Mbps was an average of a range of values from approximately 5Mbps to approximately 10Mbps, and this range was not notably impacted by speed. Note that across all experiments, the reported transfer rates never exceeded 10Mbps compared to the theoretically maximum possible of 27Mbps. The loss rates, even at very short distances (near 0m), were considerable at 5%, becoming higher (reaching and exceeding 20%) for longer distances (100-150m).

The use of experimental testbeds, such as the ones used in [4, 19, 51], to collect realistic measurements is a challenging task due to the investment in time and equipment, and are producing results that are not easy to claim as generalizable. In trying to understand the factors influencing the performance, some, like Bloessl et al. [6], have put emphasis in digging deeper into the physical layer dynamics, which led them to the implementation of IEEE 802.11p transceivers using Software Defined Radios (SDRs) but the experimental apparatus becomes costly and cumbersome to deploy. Nevertheless, the vast majority of the performance studies of IEEE 802.11p employ simulations or analytical tools, resorting to various approximations. For example, one notable attempt to capture the reality of non-homogenous vehicular traffic is introduced in [40] using a fluid density model to capture the interaction of the vehicles via density-dependent velocity. Such a model was found to map

well vehicular traffic interactions. e.g., with traffic lights. The model was able to produce location-dependent throughput and delay in agreement with simulation results. This came at the cost of modeling just periodic broadcast beacon messages, with the authors acknowledging that the performance would deteriorate further once data traffic other than beacons was introduced.

It is not surprising that in trying to improve the IEEE 802.11p performance, researchers are increasingly looking at combinations with other protocols. For example, Wu et al. remark in [57] that *"... we believe that a heterogeneous system with 802.11p and Sub-1GHz channel can outperform the single 802.11p system in V2V communication."* and proceed to use an extra sub-GHz channel to attaining better performance than IEEE 802.11p in urban settings. What improves the resilience of the solution in [57] is the fact that the lower frequencies used in the Sub-1GHz module (433 MHz in [57]) are better suited for longer range communication. But if combinations of protocols are to be considered, why not consider an obvious one: DSRC and cellular. Indeed, Ligo et al. [28] report on a network prioritization strategy, of On-Board Units (OBUs) sending preferably via DSRC with fallback to cellular Long-Term Evolution (LTE), in a study informed by mobility trace data from the Portuguese city of Porto. Their main conclusion was that the cost-benefit analysis does not favor the deployment of DSRC for the purposes of data connectivity to the Internet. Whether explicit or implicit, the cost-benefit assessment may have been a factor taken into account resulting in poor DSRC adoption.

2.1.2 The Failure of IEEE 802.11p

The reader should notice that at the time of writing of this thesis (2023), several years have passed since the introduction of IEEE 802.11p and one would expect a flourishing of associated products and services, at least if we are to judge by the excitement of years past in research publications. This is not the case. Some of the lackluster performance regarding the adoption of the protocol was due to widely used cellular alternatives for data services that can cover well (but for a fee) the data needs for non-critical applications. In the meantime, cellular services have made significant inroads towards supporting critical services, as, e.g., 5G, promises very low latencies. However, a recent,

regulatory, development put a stop to DSRC altogether.

In 2020, the Federal Communications Commission (FCC) adopted [16] *"new rules for the 5.9 GHz band (5.850-5.925 GHz) to make new spectrum available for unlicensed uses, such as WiFi, and improve automotive safety."* which effectively introduced two changes. One change was that spectrum was taken away from the band reserved for DSRC and given to WiFi. The reason was *"DSRC has not been meaningfully deployed, and this critical mid-band spectrum has largely been unused for decades"* thus, not only DSRC lost, but WiFi clearly won. The same FCC announcement replaces DSRC with a cellular alternative to Intelligent Transportation Systems, which is commonly called Cellular V2X (C-V2X). C-V2X is a 3rd Generation Partnership Project (3GPP) standard and while in principle it can be a standalone protocol, it greatly benefits from being tied to other regular cellular services, e.g., with respect to synchronization. The FCC decision concedes the Intelligent Transport Systems (ITS) to cellular providers and equipment manufacturers. While the research on how well C-V2X can support vehicular services is still ongoing, some evidence suggests C-V2X might be superior [32] to DSRC. The application examples provided are status broadcasts, where vehicles provide their identity, location, speed, etc. at regular intervals. No mention is made about bulk data transfers support, possibly implying that those have the regular cellular data services to rely on.

2.2 The WiFi Alternative

It has been almost two decades since the first experimental studies for the use of WiFi in vehicular networks were published and to put forward ideas of opportunistic communication. Ott and Kutscher's study, [37] described the "Drive-Thru Internet" using the IEEE 802.11b protocol with APs at fixed locations and a vehicle driving by at highway speeds: 80km/h and 120-180km/h. Their measurements demonstrated that, as a vehicle drives by an AP, even at 180 km/h, they could obtain in the order of ten seconds connectivity. The pattern of the connectivity follows a three phase pattern: while approaching the connectivity is poor but improving, followed by a plateau while passing near the AP, followed by a deteriorating phase as it moves away from the AP.

TCP throughput performance was found to be peaking at 3Mbps to 4.5Mbps depending on speed. They report that the best they could deliver in a single pass by the AP was nine megabytes of data, although at the highest speed (180km/h) this was around 1.5 megabytes. The packet loss results for UDP varied across a wide range, with the worst losses being in the 80% range. Note that, it is expected that IEEE 802.11 transceivers perform their own rate adaptation by selecting the modulation and coding scheme to match the channel condition, and the packet is lost when the corresponding link layer frame has been re-transmitted a few times at possibly different rates. The same group extended their study to IEEE 802.11g, [36], which allowed higher bit rates and a corresponding improvement in attainable transfer volumes per drive-by encounter with an AP.

A study by Bychkovsky et al. [7], using IEEE 802.11b transceivers on nine vehicles considers a real urban environment, involving speeds no more than 60 km/h, over a period of a year. Technically, they mimic a V2I scenario where the infrastructure are APs that require no authentication. The median connectivity interval to an AP lasted 13 seconds and an AP would be encountered, and associated with, on average every 75 seconds. The median upload capacity for TCP was 0.24Mbps (30 kilobytes per second) and a median per-encounter upload capacity of 216 kilobytes. The transfer speed was limited by the a-priori decision of the researchers to limit the physical bit rate to its lowest possible of 1Mbps, which, coupled with the use of the maximum transmission power of 200 mW, provided the longest range. Bychkovsky et al. made a number of insightful observations on the impact of AP discovery, using scanning, and the association to the APs. The impact is that time is spent during the beginning of the encounter between vehicle and AP, and while the link conditions are still not good, before actual data transfers can commence.

We remark that the motivation behind the Bychkovsky et al. study was the existence of many "open," i.e., requiring no authentication, APs at the time the study was conducted. Alternatively, it assumes a degree of volunteerism to ensure a good number of open APs exist. In recent years, APs requiring no authentication are rare, and when they are available, they expect interaction of the user after they have associated with the AP, e.g., by agreeing to terms

of service on a captive web portal. The process described in [7] involving a `ping` packet to a server after AP association would have failed in almost all open APs today. Moreover, in order to obtain an IPv4 address, they use the Dynamic Host Configuration Protocol (DHCP), and to improve the delays involved in address lease, they implement an address caching scheme. Overall, the connection establishment time was found to have a significant impact on performance.

A subsequent careful and insightful study by Hadaller et al. [22] presented what was achievable in terms of data transfer capability between an (on-vehicle) OBU traveling at 80km/h and a Roadside Unit (RSU) communicating using IEEE 802.11 protocols. Incidentally, Hadaller et al. did not use the terms OBU or RSU in their paper, as the terms were not in wide use at that time. However, they effectively described the communication needs of devices that, in today's terminology, would equal OBUs and RSUs. The experiments reported in [22] highlighted the complicated interplay that protocols and devices have on the observed performance of such a system, given the transient and volatile nature of the communication "links" established between client devices (OBUs) and APs (the RSU was operating as an AP). Their study is also notable for the effort put into overcoming the GPS unit's lag, infrequent location sampling, and noisy location estimates, for determining the exact coordinates of the vehicle when each frame was transmitted. The setup allowed to create accurate maps of distance from AP versus three metrics of interest: the signal strength, the bit rate at the MAC/PHY layer, and the TCP goodput. To the best of our knowledge, they are the first to look at the rate selection being performed by the IEEE 802.11g transceivers, which they found to be poorly performing when the default behavior was used. Because of the particular transceiver and device driver, they were able to modify the rate adaptation scheme and to improve performance. The average bit rate when close to the AP reached 50Mbps, for a IEEE 802.11g maximum rate of 54Mbps, with the TCP goodput reaching an average of approximately 22Mbps, and never significantly exceeding 30Mbps. Similar to other previous work, they noticed the three phase behavior of performance as the vehicle moves by the AP. Similar concerns to those by Bychkovsky et al. [7] were raised regard-

ing initialization (DHCP, Address Resolution Protocol (ARP), etc.) specific to the IPv4 network layer.

In yet another study, Cottingham et al. [9] employed IEEE 802.11a, which was the first generation of WiFi protocols in the 5GHz Industrial, Science, and Medicine (ISM) band, and a testbed involving a moving vehicle and a fixed AP. The speeds used were 7km/h and 45km/h, justifying the lower speed as a congested city bumper-to-bumper traffic. The tests were exclusively using UDP with traffic demands of either 10Mbps or 30Mbps, having excluded TCP due to the impact they expected the high loss rated would have on its performance. While both of the maximum rates of 10Mbps and 30Mbps were achieved, there were a number of lower rates when the demand was 30Mbps while when it was 10Mbps, it appeared the rate was "binary" either achieved or not at all. One interesting finding was that while travelling at lower speeds results in longer connected (to the AP) time period, it also produced a more variable throughput compared to short connection periods (happening at high speeds). The explanation provided was that at 7km/h the devices would spend more time in null zones insofar their antenna coverage was concerned. During those periods in the null, the rate adaptation of WiFi probably reduced the rate. At 45km/h, nodes spend little time in the null zones. Experiments were carried that confirmed this observation.

Finally, there is not a lot of work for the ad hoc mode operation of WiFi as it relates to V2X applications. The ad hoc mode allows the endpoints to not have asymmetric roles as is the AP case (where the AP becomes a de-factor coordinator) but instead for all communicating peers to be "equals". Of note is the work by Rubinstein et al. [42] which considers two vehicles moving at the same speed from opposite directions, and relative speeds of 40, 80, and 120 km/h, on a straight road segment of 400 meters. Some other light vehicular traffic was also present on the track. The IPv4 addresses are fixed, as are the Extended Service Set Identifier (ESSID) and channel used, while the ARP entries are manually installed, all done in the interest of reducing the setup delay to a minimum. The authors report that for IEEE 802.11g, and UDP traffic, the performance depended on the packet size and speed of vehicles. The encounters ranged from approximately 11 seconds to approximately 40 seconds,

between fastest and slowest speed. At low speeds, the best transfer volume was note for large (1460 bytes) packets at an average 13 megabytes per encounter, while at high speeds it was 2.7 megabytes average per encounter for 500 byte packets – the numbers corresponded to a 2.75 Mbps and 1.81 Mbps for lowest and highest speed respectively. The packet loss rate was high at high speeds (10-20%) indicating that replacing UDP by TCP would probably be worse. Indeed, TCP at 80 km/h transferred only around 1.5 megabytes per encounter (with 500 byte packets), compared to UDP's 3.3 megabytes. At 12 km/h four out of the ten runs received no data at all using TCP. While IEEE 802.11a experiments were also attempted, they were found to be worst of all — even the best case UDP averaged 3.1 megabytes per encounter at the lowest speed. While it is somewhat puzzling that IEEE 802.11a performed that poorly, it may also have something to do with the less than ideal products used which for the particular experiments were WiFi CardBus interfaces with integrated antenna, inserted in a laptop, held by the passenger. Nevertheless, Rubinstein et al. recognized as well the impact of rate adaptation and how it should be able to respond to quickly changing channel conditions.

While followup work for use of WiFi in mobile nodes in general within the broader context of wireless ad hoc networks continued for several years, it was quickly overtaken by an interest for IEEE 802.11p, which as we described previously has now become a dead-end. The early performance studies of WiFi for vehicular applications provide us however with some valuable methodological lessons.

2.2.1 Methodological Lessons: WiFi

The present thesis is methodologically influenced by the reviewed literature in the following aspects:

- The bit rate adaptation performed at the MAC/PHY level has an impact on the ability of the IEEE 802.11 transceiver to adjust to changing channel conditions. Depending on the exact hardware and device drivers used, we may not be able to influence directly the underlying algorithm but we need to, at a minimum, track its performance over time when interpreting results. Failed transmissions, and hence poor utiliza-

tion, may be the result of poorly chosen coding by the rate adaptation algorithm.

- There are upfront costs when "connecting" with an AP, that are related to the following steps: scanning, associating, (optionally) authenticating, and establishing network layer addresses. We consider a system where those delays are reduced by means of:
  - for V2V links, by adopting a common ESSID by all fleet vehicles,
  - for V2V links, using link-local IPv6 addresses (to avoid DHCP),
  - for V2I links, "whitelisting" the ESSIDs to be used, and,
  - for all links, to only connect without any authentication.
- As we will describe in subsequent chapters, a "whitelisting" of V2I APs is possible because we do not expect that vehicles will attempt to connect to any arbitrary AP but rather to well known "rest stop" APs. With this in mind, any captive portal interaction, can be scripted and automated, but we leave the implementation of this mechanism outside the current thesis.
- We will consider the case of both AP-mediated connection as well as "ad hoc" mode communication, since ad hoc is also part of the IEEE 802.11 standard. While the literature is replete with AP based V2X attempts, there is scarce understanding of whether the ad hoc mode is up to the task, and in particular in cases where we expect all nodes to be identical in the behavior, e.g., as vehicles of a fleet.
- Finally, we would like to see how the most recent revisions of the IEEE 802.11 standard (currently the standard is IEEE 802.11ax) perform. The recent versions of the standard offer a wider set of options of bit rates, with a correspondingly enlarged search space for the rate adaptation algorithm. Additionally, implementations of recent versions of the IEEE 802.11 standard involve more than one antenna, possibly partly overcoming the shortcomings when we depend on a single antenna, e.g., the nulls of a single antenna reducing performance.

2.2.2 A Word on IEEE 802.11 Terminology

The wide use of the IEEE 802.11 standard has created a situation where various terms are employed to describe its operating principles, its components and their function. The terminology can be confusingly redundant and is a result of different user groups describing the same thing in ways natural to them. As a result, there is, (a) the language of the standard documents but also, (b), the language used by practitioners at various levels of familiarity with the technology, including casual users, the "lay public" which often, simply, refer to "WiFi", (c), the language of software developers, e.g. operating systems device driver authors and related operating system tools, who employ abstractions that are helpful for the programmer's view, and, (d), by researchers employing conceptual categories across various protocols, that are not always exactly matched to the specific IEEE 802.11 behavior.

Throughout the thesis we use a mixed terminology, drawing from the various user groups. When there is no reason to distinguish particulars, we use the generic term: *WiFi*. When we wish to distinguish a symmetric role of hosts, we name their interaction *Peer-to-Peer (P2P)* interaction, and when their role is asymmetric, it is client-server interaction, in which the server role is played by the *Access Point (AP)*. The term WiFi is often conflated with the client-server mode, as most users are only familiar with it. The client-server, AP-mediated, mode is also referred to as *infrastructure* mode. We often refer to the P2P mode as the *ad hoc* mode since it is comes close(r) to the role of ad hoc wireless network connectivity used in research literature. A more specific naming to refer to the WiFi P2P mode is to call it the *Independent Basic Service Set (IBSS)* mode, alluding to the specific way in which the P2P differs from AP mode's Basic Service Set.

Yet another naming of the role of nodes comes from the used operating system tools, e.g. `iw` in Linux. Given that the two network interfaces, the one on the client node and the one on the server (AP) node, implement partially different logic, the interfaces of client hosts to the AP are said to be in *managed* mode, which is alternatively called *station or STA* mode (as opposed to AP mode). A further state for a WiFi interface is the IBSS mode, corresponding to the definitino of IBSS, i.e., ad hoc mode node, of the previous paragraph.

2.3 (V)DTN Data Transfer Strategies

Once a communication strategy and related protocols have been selected, a DTN/VDTN can be decomposed into three other components: the application, the routing, and the mobility aspect. The application is assumed to be the broadest set possible but we hinted earlier to two main applications, better understood as belonging to two families:

1. encounter application transfers: capturing the case where the application needs a mobile agent, i.e. a vehicle, to be nearby in order to pick up data items. For example, a vehicle picks up the data from the gateway to a WSN that is not connected to the Internet, conforming to the "data mule" [45] transfer paradigm

2. proportional-to-travel applications: capturing the case where the data volume is proportional to the travel distance of the mobile agent, e.g., to express a vehicle collecting vehicle performance via OBD-II, or locations recorded via GPS, or road condition data, etc.

The main difference of the two applications is that, for the first type, data can only opportunistically be expected to be transferred from the surroundings to a vehicle, if it happens for the vehicle to drive by a sensor network gateway – which may not be always possible. In the second case, the collected data are readily available for transmission on the vehicle, possibly collected in groups, akin to files, prior to making them available for transfer by the VDTN.

Regardless of application type, a common characteristic is that either way there is a need to define a data unit of transfer, which in DTN/VDTN literature are often called "bundles" and are discussed in Section 2.4. In effect, we reduce the question about the applications to the question of bundles, and let the applications annotate bundles if they wish to have bundles differently treated. The question of routing among DTN/VDTN nodes has been studied to great lengths in the literature as we will see in Section 2.6 and it is generally intertwined with the ways that researchers evaluate, e.g., simulate, DTNs/VDTNs.

2.4 Data Bundles

The use of the term "bundle" in this thesis is consistent with existing standards, e.g. RFC 5050 [44] and it is defined as a message unit that is set up appropriately to transfer in a DTN. Bundles are supposed to need a bundle (transfer) protocol because of intermittent connectivity and because of the concept of bundle *custody* transfer which might happen upon encounters with other nodes. Given the subsequent research carried out in DTN routing which we will summarize later in this section, the RFC 5050 standard appears to be an early attempt to formalize some of the ideas on how DTN messages should be transferred, yet allowing for flexibility such as exploiting opportunistically (or in a scheduled manner) the intermittent connectivity events, and without any reference to actual IPv4 or IPv6 addresses. Beyond the concept of custody transfer and bundle deletion, RFC 5050 also introduced the idea of fragmenting bundles to cope with intermittent connectivity disallowing the complete transfer of a bundle in a single encounter. The question however is whether inventing a new protocol, such as RFC 5050, is needed at all.

Alongside the formalization of the concept of bundles, there have been attempts to formalize the DTN architecture in RFC 4838 [52]. The recommendation relies on a key assumption that we also adopt in our work, namely, storage is plentiful and well–distributed across network nodes. The availability of storage tilts the balance of decisions made in DTN nodes to favor more storage, effectively assuming it is "for free" and to prioritize the task of finding smart ways to copy and distribute the bundles to nodes they encounter. A side-effect of this attitude is that the functionality of the protocol to allow bundles to be removed from local caches of all nodes after they have been delivered need not be a high priority protocol task. Nevertheless, RFC 4838 goes to great lengths to introduce the concept and semantics of Endpoint Identifiers (EIDs) as they are not (necessarily) IP addresses but are recognized as identifiers for the originator, destination, custodian, etc. peers. At a minimum, RFC 4838 expects the following headers at the beginning of a bundle:

- creation timestamp
- lifespan

- class of service
- source EID
- destination EID
- report-to EID
- custodian EID

The creation timestamp assumes DTN nodes have a basic time synchronization capability albeit the minimum requirement may involve just a sequence number such that successively generated bundles from the same source EID follow a monotonically increasing sequence. The lifespan is an offset with respect to the creation timestamp beyond which the message is to be discarded, implying that the creation timestamp sequence has at least some semblance to natural time, e.g., seconds since the originator node started operating. The class of service are a form of prioritization. The report-to EID captures the situations where nodes other than source EID need to be informed of the delivery and, absent any other information, report-to EID is to be usually the source EID. The custodian EID is the address of the current custodian node (if any).

In a confusing statement in RFC 4838 *"The EID, timestamp, and data offset/length information together uniquely identify a bundle."* which would suggest that there is a sequence number referring to individual bytes originating from an EID, such that when EID and timestamp (due to its resolution) are the same, there is at least a data offset/length information that is unique. A more reasonable assumption for a unique identifier could be the hash value of the contents (inclusive of EID, timestamp, and any other fields deemed essential), allowing the specification of a unique identifer of fixed length (the hash value) regardless of the length decisions related to the EID, timestamp, or any other field for that matter.

What neither RFC 5050 nor RFC 4838 describe in detail are any mechanisms for how exchanges, priorities, policies of retention, etc. are to be implemented. This is unsurprising given the need to keep, at the time of publishing, all options open. RFC 7242 [11] defined a layer on top of a reliable transport layer, namely TCP, to support DTN transfers, which they called the

TCP Convergence Layer (TCPCL). Even then, TCPCL is restricted to essentially a fragmentation logic of bundles in possibly multiple data segments. The negotiation stage of TCPCL also caters to issues related to fragmentation decisions. RFC 7242 is also reluctant to describe the semantics of the exchange, e.g., policies of how many copies of a bundle may be made, if aging of a bundle should influence how urgently nodes need to send it to contacted nodes, etc. All such concerns are left to the application layer logic. In short, the standardization attempts have focused more on the syntax and simple(r) transfer-related questions, like fragmentation, rather than the core DTN logic. This may be an indicator that all DTNs are "one of" solutions, resisting many attempts to extract a common, and useful to all, functionality to provide as a protocol layer.

2.5 The e-mail Alternative

The position taken in this thesis is that it is largely unnecessary, given the existence of other protocols, and, in particular, those related to e-mail, to indulge in complicated bundle definition and bundle-related protocols. In fact, e-mail message formats already provide a sufficiently large set of options that meet and exceed what the bundles in DTNs need. A head-to-head comparison on key similarities and differences, with a subjective indicator whether email has an advantage, disadvantage, or is even, compared to bundles is as follows:

1. *Header Variety* (advantage): E-mail headers are some of the most comprehensive headers across any protocol currently in existence [23, 38]. A DTN implementer has the flexibility to use legacy fields like the `From:` and `To:` to identify source and destination EIDs. E-mail addresses can be either meaningful as actual Simple Mail Transfer Protocol (SMTP) recipients (in addition to being legitimate DTN EID identifiers), or be exclusively meaningful for the DTN alone. If there is no already defined similar header to a header the DTN logic requires, the option for using extension, `X-` prefix, headers always exists. As an example, if the DTN logic confers the permission to each node to make up to a maximum number of copies of a message, it could introduce a (mutable) header

field such as `X-Copies-Allowed:` followed by a numerical value.

2. *Opaque Data Transfer* (even): E-mail is used to convey any number of file types and attachments without any specific regard to their content. Normally the e-mail payloads are immutable. Some of the headers may also be immutable. The representation of the data can be flexibly described in the headers (`Content-Transfer-Encoding`) as being textual or binary, or even user-specific. As such, an e-mail message, beyond the headers, is a self-contained bundle of data and, if desired, it can be up to the endpoints alone to attach any particular meaning to the contents.

3. *Legacy Code Base* (even): There are both e-mail client and e-mail server implementation with large code base which can be repurposed to parse and handle messages with various headers. Compared to writing code from-scratch for a bundle transfer protocol, there are some possible effort savings to be had, plus the possibility that e-mail related code has been tested more thoroughly. The common disadvantage for both from-scratch and e-mail-based implementations is that neither have a complete implementation of DTN bundle handling policies, etc. The effort saved by using e-mail specific header and body parsing modules may not be as impressive compared to the code that will have to be written from-scratch.

4. *Security Options* (advantage): Whereas bundle and DTN protocol standards defer the discussion of security options for future, evolved, standards, e-mail already has a well-tested solution in Secure Multipurpose Internet Mail Extensions (S/MIME) (currently in version 4.0 [43]) which provides confidentiality, integrity, authentication, non-repudiation with proof of origin, as well (optionally) compression. The choice exists [8] to protect the payload plus certain headers as one cryptographically protected object, declaring them as a `message/rfc822` MIME object wrapped in a cryptographic envelope.

5. *Message Length* (disadvantage): A notable disadvantage of e-mail is that messages are restricted in terms of maximum length to a few tens of

megabytes. Bundle protocols seem to have looked into fragmentation from the very beginning. Nevertheless, fragmentation comes with the cost of re-assembly and committing storage for that reason. Insofar the e-mail size limitation is concerned, it is largely a made up limitations arising from the Mail Transfer Agents (MTAs) wishing to regulate the storage they need to commit and the duration of transfer for a single messages. When we reuse legacy software, we can certainly relax such size restriction. However, in light of the typical volumes in V2V transfers in Section 2.2, in order to have the opportunity to transact in the order of a few bundles per encounter, the sizes of those bundles cannot be more than at few megabytes each. We propose that this is addressed as an engineering constraint placed *on the generation* of the bundles, i.e., make bundles a few megabytes large. The constraint may not make sense in other DTNs but is meaningful for our VDTN case study.

6. *Fallback Protocol* (advantage): A special-purpose bundle protocol is an isolated protocol from the rest of the Internet and its format may be incompatible with any other protocol. However, if we start, to begin with, with a proper (including essential headers) e-mail envelope, e.g., a meaningful `To:` email address, then if the message arrives to a node that has access (even if spotty) to the Internet, it can deliver the message to a destination server using SMTP or Internet Message Access Protocol (IMAP) while also performing on the DTN side according to the DTN logic. That is, any inbetween node is a potential exit to the Internet without message format changes.

Based on the above points, we consider the need to develop a custom bundle format and protocol as a secondary issue and meaningful only in cases where one wishes to create a low implementation complexity proof-of-concept. If one wishes to develop a more general solution, then the challenges have already been solved in the design of e-mail message transfers, including endpoint identifiers, custom headers, disposition, payload flexibility, and security. Thus far our discussion has been more syntactically-oriented. We complete the related work section by looking at how (V)DTNs are evaluated with respect to routing decisions.

2.6 Routing

What is left outside the scope of standards is the exact process of deciding how data bundles will be routed as they are relayed among nodes. Custody is only one of the relevant concepts. One possibility is to copy a data bundle to any vehicle encountered that does not already possess the particular data bundle. On the surface, this strategy appears wasteful. Yet, storage is plentiful today and the strategy maximizes the odds that at least one vehicle carrying a copy of a bundle will reach a facility, e.g., an open AP, to deliver the bundle to some server on the Internet. This is a variety of the, so called, *epidemic routing* but there are more options.

A recent paper on VDTN by Er at al. [15], names a set of routing protocols as "baseline" protocols, because they form the basis for numerous variations and extensions. The research area of routing for DTNs/VDTNs is already containing numerous proposals and it is not the main focus of this thesis. There exists a sufficient variety of routing protocols to allow an application designer to balance performance, such as average delay, copies of a bundle circulating, throughput, etc. A simple, epidemic, routing strategy is incorporated in the proof-of-concept implementation described in the next chapter as a matter of demonstration, but others could have been used in its place. The baseline VDTN routing protocols are [15]:

1. *Direct delivery*: according to which, a mobile, e.g., vehicular, node delivers the data directly to its destination when it happens to be, during its trips, within communication distance of the destination. This definition has been used, e.g., in [46], but one can define a slightly broader class of similar protocols. Namely, as long as we assume that a single copy of a bundle exists in the network, its custody can be transferred to another encountered node, if the odds of that node performing direct delivery are better than the current custodian. That is, direct delivery can be the last step of a process of relaying to a single node the responsibility of delivering a data bundle. The approach presumes that upon encounter with another node, the current custodian possesses some information allowing it to decide whether to retain custody or to pass it to the encountered

node. To this end, Jain et al, [26] exploit knowledge about the topology dynamics, even if such knowledge is imprecise and time-varying. The nodes attempt to follow a "good" path of transfers to the destination. In the absence of any better node choice to pass the bundle to, a node has no other choice but direct delivery.

2. *First contact*: according to which no assessment of the suitability of the encountered node is made and a node just passes the bundle to the first node it encounters. The receiving node behaves the same way, thus resulting in the bundle appearing as it is performing a "random walk." While only one copy of the bundle is in circulation at any point in time, its custody is transferred continuously. The intuition supporting this approach is that a constantly "mobile" bundle hedges the mobility to increase its chances to, eventually, be held by a node in close proximity to the destination. The cost is that, especially when encounters are rare, the node picking up the bundle may be moving in a completely counterproductive direction to the desired destination. Both direct delivery and first contact operate on the implicit assumption that multiple copies of a bundle are undesirable, but the current abundance of inexpensive storage makes this an unnecessary concern.

3. *Epidemic routing*: whereby copies of each bundle are forwarded to each node encountered, assuming the encounter duration allows for such an exchange to take place. At the cost of the storage and the eventual need to purge copies of a delivered bundle, epidemic routing maximizes the odds to have a bundle delivered quickly. To curtail the, potentially massive, overhead of a complete flooding, works such as [54] propose limits on the hop count of the bundle travel. Another way to limit the extent of flooding is, for example, the technique employed by PRoPHET [29], according to which a node retains its copy of the bundle but also copies it to another node, if that node has a more favorable metric, giving it a higher chance to encounter the destination. Additional constraints need to be applied [54, 29] when the amount of node storage is limited. Such limits also require the introduction of policies to decide which of many

stored bundles needs to be dropped first to accommodate new bundle(s).

4. *Spray & wait*: whereby the behavior of the nodes can be seen, for each bundle, as split into two phases: a first flooding stage, followed by a direct delivery phase. The flooding is limited by bounding the number of copies that can be simultaneously circulating. The strategy, as described in [46], starts with a budget L of copies which is a permission to have up to L copies in circulation. In its "binary" mode, it involves splitting evenly the budget of copies to an encountered node, and the encountered node splits the budget further across the nodes it encounters, and so on. Effectively, for as long as a node has a budget of more than one, it is allowed to "infect" with a copy (and pass half its budget) to an encountered node that did not have the bundle. This aggressive copying phase is followed by the phase of direct delivery. The second phase arises naturally without explicit synchronization because, through the splitting of the budget, the nodes with a copy will eventually have a budget of one and a copy of the bundle.

A significant body of work has already accumulated on the performance study of various DTN and VDTN routing schemes. There are many possible mobility scenarios, several factors that could be taken into account (e.g., storage limits), and various metrics that are considered in these studies. For example, Abderlkader et al. [1] find, unsurprisingly, that protocols that try to guide the selection of hops, such as PRoPHET outperform, in terms of delivery ratio, the indiscriminate routing. However, the delivery ratio is not always without a penalty in the statistics of delay until delivery if there is no way to prioritize among bundles. Protocols such a Spray & Wait offer a reasonable compromise.

2.6.1 Routing for VDTNs

The general attitude in the performance study of DTNs such as in [1], is to consider destinations that are arbitrary nodes, rather than some "exit points" to the global Internet. One could argue that in many VDTN applications, except for strictly V2V, the eventual destination is a computational and stor-

age facility in the Internet, possibly in the cloud. The reverse direction, from Internet to vehicles is also a possibility, e.g., in an application that pushes firmware updates to (some) vehicle computers. A precise definition of entry and exit points from/to the Internet is also an often neglected scenario in general DTNs but part of VDTNs in V2I scenarios.

Pereira et al. [39] explain how DTNs map to VDTNs and provide a survey of various facets of the routing problem, without offering any new quantitative performance comparison. It is worth noting that the insistence on bundles between a source and a destination among the vehicles, reveals a V2V emphasis. V2V could be mapped to two V2I phases, one from vehicle to infrastructure and one from infrastructure to vehicle. This might sound as doubly penalizing in terms of delay any V2V communication, but if the intention was to provide a (near) real-time V2V, then *any* DTN approach is by definition unsuitable. It is difficult to imagine niche applications where V2V is essential but real-time deliver is not.

Tornell et al. [53] produce a survey which included the impressive number of more than forty DTN/VDTN routing/dissemination protocols. They belong to the minority of researchers that noticed that the V2V application (as an instance of Peer-to-Peer (P2P)) applications ought to be just one use case – which as we mentioned is difficult to defend if delay tolerance is expected. They note additional application classes like: (a) V2I applications performing, e.g., environment-sensing applications with the vehicles becoming the collectors of the information before communicated to a RSU, (b) dissemination of information in the sense of broadcast and multicast, where adding epidemic routing becomes a sensible choice, and (c) cooperative download from the RSU to the vehicles (or a subset of them), e.g. firmware downloads as we noted earlier.

In the more recent survey Er et al. [15], using the ONE simulator [27] and a map of Helsinki, compare direct delivery, first contact, spray & wait, and epidemic routing. The particular contribution is that the vehicle density influences the relative performance of the four schemes insofar the delivery delay is concerned. As expected, epidemic routing performs very well but what is also evident is that at low vehicle densities, the delay differences among the

four schemes are almost non-existent. We note that in a realistic setting the vehicle density used in studies does not agree to the actual vehicular density because the studies consider all vehicles to be participating in the protocol. The number of vehicles participating in a protocol may only be a small subset of all the vehicles on the road. Yet, the vehicular movement dynamics are influenced by the total set of vehicles regardless of whether they are participating in the protocol or not.

2.6.2 Simulating VDTNs as Opportunistic Networks

Simulating Opportunistic Networks: Survey and Future Directions by Dede et al., [10] is a very comprehensive study of the facets of Opportunistic Networks (OppNets) one is confronted with when setting up simulations for performance evaluation purposes. OppNets and (V)DTNs are very similar concepts, whose boundaries are blurred and we will not attempt to tease them apart. Suffice is to say that the "purist" view of OppNets is that they are networks whose function relies on the occasional existence of direct, localized communication, instead of relying on any infrastructure, and hence can operate anywhere. OppNets do not have any prescribed intent for them to connect to the broader Internet in order to accomplish their reason of existence. In short, OppNets are their own, closed, networking world.

Among the observations by Dede et al. was that the, at the time, available network simulators (ONE [27], OMNeT++ [35], ns-3 [34], Adyton [33]) were not fully capturing either the needed communication layer technologies or wireless environment simulations, or they did in a very patchy manner. Nevertheless, with their shortcomings taken into account, OMNet++ and Adyton were found to be the most efficient in terms of processing time. Also, there are several mobility-oriented tools, such as, BonnMotion [3], Legion [5], PedSim [41], or SUMO [13], which range from capturing simple random mobility models, to quite detailed trajectories on road network maps. While there is realism to having a map-referenced population of vehicle trajectories (including the effect of speed limits, traffic lights, variable vehicle populations etc.) it requires effort to set up and their results are difficult to generalize in other map layouts. Real traces from actual deployments are rare, generally useful,

but also inflexible when it comes to generalization of their results.

When they come up with some guidelines, a rather surprising one is the suggestion not to bother much with radio propagation and interference models. They suggest to evaluate them on testbeds or some outsourced computation intensive platform. While surprising on the surface, their recommendation seems to be motivated by the fact that, for the increased relative effort of including a radio propagation and interference model, the cost to be paid is large, especially computational, for possibly small performance differences. The attempt to simplify propagation models leads us to assumptions like the "binary" assumption that a transmission is received up to a certain distance, and no further giving rise to a Unit Disk Graph (UDG) model. Traces on the other hand are very specific in allowing us to determine the instant a contact started and when it ended.

2.6.3 Methodological Lessons: VDTN Performance

Given the vast literature in the area of DTN/VDTN routing and data dissemination, we make the following decisions regarding the methodology followed in the thesis:

- We will not introduce yet-another VDTN data (bundle) routing policy. There are several proposals already and their relative performance is generally understood as well as their tradeoffs. However we favor the epidemic routing family as we consider the storage as an unnecessary limitation for today's storage technology capabilities. Based on the earlier results on WiFi performance, we expect that the bottleneck is the communication: a node is unlikely to be overwhelmed by copied bundles of other nodes given the encounter times are expected to be short, and hence the transfer sizes to be small.

- For a fleet of vehicles where each vehicle has to follow a sequence of visits which is different for each day, the end of the working day defines the largest possible delay data may experience. This is because we can trivially assume that anything not delivered earlier via the VDTN will be delivered through the fleet infrastructure at the depot where the

vehicles return. As a result, a common concern of VDTNs, that of delivery delay, is essentially capped. This also means that the bundle does not have an infinite sojourn in the vehicle storage cache as they will be eventually delivered at the end of the day. Note that the fleet vehicle platform also means that we do not have to consider energy as a constraint.

- The communication model we adopt is an improvement over the UDG, making use of the earlier observations about the distance-dependent achievable bit rate. We consider a set of concentric circles, defined at critical ranges from each node, each distance "ring" corresponding to a different expected throughput. Hence, while an encounter can start from a far distance, its throughput at that point is low, getting higher as the distance decreases. By adopting this model, we can also fit testbed measurements which provide throughput vs. distance relation for various environments. The compromise is that, on one hand the coverage appears idealized (the circles), but on the other hand, we capture the progression seen in the WiFi performance works mentioned in Section 2.2, i.e., from low bit rates and throughput to higher ones, and eventually back to lower ones as two nodes move apart.

- We use the SUMO [13] simulator for the vehicular mobility. A side-contribution of the thesis is a description of a process pipeline (currently involving some manual steps) to create a realistic fleet management scenario for a given map where the fleet is supposed to operate. We consider elucidating this process as more valuable than a quantitative study tied to a single map. The purpose of the simulator is to extract helpful encounter statistics, i.e, statistics about the various distance ranges that influence the bit rates achieved throughout the encounters. In particular, we track the conditional probabilities of moving from one distance range to its adjacent (either closer or further) distance ranges.

- The set of applications we consider have already been outlined in Section 1.5, i.e. primarily V2I with volume proportional to the trip duration or encounter-based (for data mule role), thus defining a very basic

but flexible data demand model. As noted, V2V applications are probably not a natural fit to VDTNs given their expectations to be (near) real-time. We do not make explicit whether unicast or broadcast (or multicast) applications are considered. However, to the extent that it is unicast, it is a transfer from a vehicle (or the WSN from which we picked from data mule purposes) to a node (server) on the Internet. If it is broadcast or multicast, the assumption is that epidemic routing that we use can be utilized for that purpose as well.

- The applications we consider are limited to fleets whose total number of vehicles is small compared to the vehicles found in the region they serve, e.g., the total number moving vehicles of a city. Only the vehicles of the same fleet are interested to form a VDTN among them, although in the same region there may be more than one fleet, each with their own VDTN. Hence, overall the density of the vehicles of a single fleet/VDTN is very low and that can have an impact on the relative performance, as e.g., was pointed out in [15] insofar the delay was concerned. Given that the delay is also bounded as we just explained, we consider it a less interesting performance metric compared to the statistics of encounters and the volume of data we expect to transfer during such encounters.

Finally, an obvious suggestion could be: if the trajectories of the vehicles departing a depot are (more or less) known upon departing when they leave the depot, why is it not possible to compute in-advance when they will meet. While this is enticing as an idea in a fully deterministic environment, there are at least two major sources of unpredictable delays that quickly extinguish any excitement for a-priori calculation of encounters: traffic delays and unpredictable sojourn times at the service points. Almost contrary to the desire for a-priori calculation is the actual opportunities of e.g., vehicles that were not likely to encounter each other, to do so due to traffic slowdown. Not only is the deterministic scenario an unlikely outcome, it is the randomness of delays that creates new possibilities for unanticipated encounters.

Chapter 3

Proof-of-Concept Implementation

In this chapter we go over the example VDTN system, as described in the introductory chapter, which has been prototyped and tested using commodity hardware and a combination of free, as well as custom software we have developed. The goal of this system is to extract and share information between vehicles in an affordable yet efficient way. The extraction and bundling of data, along with subsequent transmission of these bundles with neighboring vehicles is also covered. The software components of our system fall into two logical categories: (1) Data Acquisition (the gathering of data through various sensors) and (2) Data Transfers (the VDTN portion of our system). Figure 3.1 offers a block diagram view of the software components and their interactions. All tests and prototypes were built on top of the Linux operating system. It is important to note that the hardware choices made for our prototype are not resolute. Decisions were made based on affordability, accessibility, and practicality criteria. A relatively short conference paper, [50], presented some early work towards the proof–of–concept implementation described in detail here.

3.1 Hardware Platform

A first set of flexible prototypes were built for testing purposes and evolved over design iterations. Their hardware components can be seen in Appendix B. The first set of prototypes used Raspberry Pi (RPi) as the computating platform, working at the heart of the OBU. A first set of tests were conducted using

several RPi 3 and RPi Zero W boards, although only the latter of is featured in the photos shown in the Appendix. In general, RPi was chosen because they run the Linux operating system, are compact in size, easily powered, affordable, and have high storage capacity (using micro SD cards). Their computing power was found adequate for the purposes of the project but this is an outcome of the mainly event-driven behavior of the platform, where CPU-intensive tasks, like compression, are invoked infrequently and can run in the background. The plethora of RPi pins (both general purpose and special purpose) allow for easy communication with external data-gathering sensors, which is crucial for building an OBU.

Three types of sensors were chosen for our testing. They represent a baseline of disparate types of data that we believe would be of interest to most scenarios involving any type of fleet management: (1) location data of the vehicle through GPS receivers, (2) vehicle Controller Area Network (CAN bus) data made available via an OBD-II port, including vehicle speed (km/h), engine speed (rpm), throttle position, engine coolant temperature, oxygen sensors, etc., and (3) acceleration data extracted in real-time from an accelerometer module connected to the RPi. The accelerometer can provide more accurate and reliable acceleration/deceleration information, as well as possible diagnostic information in case of collisions with vehicles or other objects. For example, our system allows for various software plugins that could, if one wishes, alert via cellular communication in the case a vehicle has decelerated too quickly or detects a vehicle roll-over.

3.2 OBU Software Behavior

At the highest level of abstraction, and in simplest terms, each vehicle attempts to do the following:

- Gather data that is relevant to the vehicle/driver. i.e., where am I?, How fast am I going?, Is the vehicle being operated responsibly?
- Every so often, bundle all acquired data. Bundling includes encryption and compression. In this sense, a bundle represents all accrued data

from a slice of time (see “proportional-to-travel” applications in Section 2.3).

- If within transmission range of another participating vehicle or WSN sink, share as much data as possible during the encounter.
- If we come within transmission range of a designated/open WiFi AP, upload as much data as possible, as efficiently as possible, to a central server shared among all vehicles.

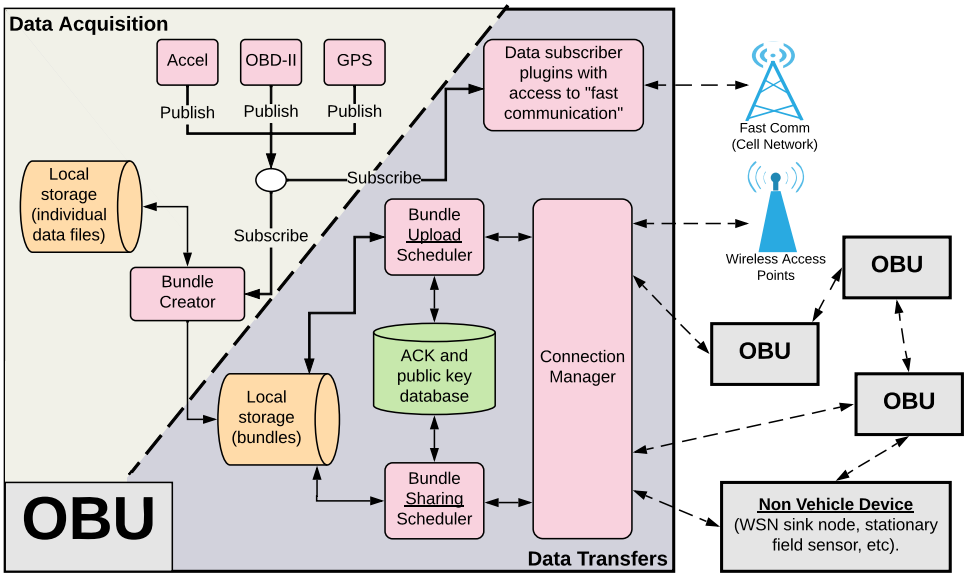


Figure 3.1: Interactions between software components. The components have been separated into two general categories: Data Acquisition (top left) and Data Transfers (bottom right)

Figure 3.1 provides an overview of the relationship between our software components. The separation via dotted diagonal line is purely logical and meant to more easily visualize the difference between acquisition and transfer of data. Beginning the whole process is any number of programs needed to extract useful data from sensors. In our example, these can be observed as *Accel*, *OBD-II*, and *GPS*. We use a publish-subscribe methodology to illustrate the way in which this acquired data is published to a *Bundle Creator* which

encapsulates all data into manageable bundles. Disparate data can be continuously written to storage in temporary files, and ultimately as completed bundles. This concludes the summary of data acquisition within the OBU.

3.2.1 Data Acquisition & Bundling

There exist two general steps to data management in our system: extracting data (from sensors) and bundling that same data afterwards. This allows for considerable flexibility in terms of adding and removing extraction programs. The only requirement from such a program is that it must subscribe to the *Bundle Creator* and publish its data in whichever format it desires. Typically, it will do so at regular intervals, but ultimately it is up to each extraction/acquisition program to control when and how they publish their data.

When it comes to packaging our acquired data into more manageable units we have chosen to create and implement a very specific bundling mechanism that sits at the highest layers of abstraction and is network-agnostic at its core. Contrary to much of the research that has been focused on developing more generic bundling and routing protocols for opportunistic and delay tolerant networks, we have chosen to move all bundling and routing decisions to the application layer. The *Bundle Creator* only cares about creating bundles and storing them on disk for the schedulers to handle.

As shown in Figure 3.1, a publish-subscribe messaging system for extracting and managing sensor data has been adopted using *ZeroMq* [59]. Each sensor module has an accompanying process written in `python` (Accelerometer, OBD-II, GPS) that reads data from its associated sensor, prepends timestamps to individual data, and publishes it for reception by a *Bundle Creator* and (optional) *Fast Comm* data plugins. The plugins can be added as needed and may be tailored to handle only a subset of the disparate sensor data being acquired by the OBU. Fundamentally, *Fast Comm* can handle delivery of time critical messages via a, typically, continuously available but expensive communication link such as cellular networks. The plugins can trigger the emission of such messages either as regularly scheduled notifications of raw sensor data like GPS coordinates, or customized messages triggered by events such as breaching a geo-fencing boundary, or sensor evidence of vehicle mal-

function and/or accident.

The *Bundle Creator* continuously appends the received data to separate sensor-specific temporary files on the local filesystem. At specified (periodic) time-intervals, it bundles all data contained within these temporary files using MIME encoding, and stores the newly created bundle back on the local filesystem. At that point, the temporary files that were being appended to are closed, and new ones are opened for the process to continue appending data without interruption. In this way, each bundle represents a snapshot of all sensor data gathered over a particular interval.

The bundling process itself is shown in Figure 3.2. It starts with the retrieval of all temporary data-acquisition files from the local filesystem. Each of the n individual data files, *Data_1*, *Data_2*, ..., *Data_n*, is then compressed using *gzip* and incorporated into a multi-part envelope using MIME encoding. Three pieces of metadata are included in the header of this temporary envelope:

1. **Intended Recipient:** the final destination of the bundle, i.e., an identifier for the backend server.

2. **Sequence Number:** used for bookkeeping purposes.

3. **Time Validity:** signifying the soundness of the timestamps contained within the individual data files.

With respect to time validity, it should be noted that platforms like RPi do not include a persistent real-time clock. The timestamps therefore depend on the options available to each platform, and one has to be aware of the origin of the timestamps for OBUs or for sensor data bundles. We introduce **Time Validity** as a marker of the source of timestamp information used for the data records in a bundle. Currently, we have defined the values `GPS`, `RTC`, `STORED`, and `NONE` to indicate whether the bundle's timestamps are referenced to GPS time, to an add-on Real Time Clock (RTC) module, to time stored at shutdown and restored on reboot, and (`NONE`) to local time that is always reset upon reboot. Combining this information with sequence numbers (that are assumed

to be always monotonically increasing and be persistent across reboots) allows post-facto corrections to be performed to the timestamps, depending on the backend application requirements.

The envelope is then digitally signed with the device's private key, and this signature is added to the contents of the current envelope. The signature facilitates non-repudiation of the bundle contents with respect to the device that produced it. At this point, the plaintext bundle envelope is considered to be in *server* format. To convert the bundle into a more sharable version, the current contents need to be encrypted in order to protect the privacy of the creating device. This is accomplished by encrypting the entire bundle contents (header, compressed data files, and signature) using the back-end server's public key. An option that exists at this point, and which we incorporate, is to consider whether the encoding of the object is transferable via text-oriented (7-bit ASCII) communication channels. If this is the case, it is prudent to convert it to base64 encoding.

In order to provide peer devices with enough information to properly categorize and prioritize these sharable bundles at the *Data Sharing* stage, new plaintext metadata must be added. In particular, the device's identifier and bundle sequence number are included, along with another signature for integrity checking to ensure that the bundle has not been tampered with. OBUs can then proactively verify the contents of this sharable bundle (new header and encrypted bundle envelope contents), by using the public key corresponding to the device identifier. The distribution of the public keys of a fleet's devices can be part of the communication to the backend servers, alongside the transfers explained in the next section. In principle though, a device may carry a bundle of unverified origin for some time before it is able to verify it. Even if the OBU cannot verify it, the verification and integrity check can be deferred to the point when it is received by the backend server, which also uses these keys to verify the signature of the inner bundle envelope, after decrypting it. Lastly, the bundle is named using a globally unique ID which is the hash value (currently MD5) of its contents. The bundles are effectively treated as unique files, and their upload and sharing described subsequently can rely only on their MD5 hash IDs, even though as we will see the schedul-

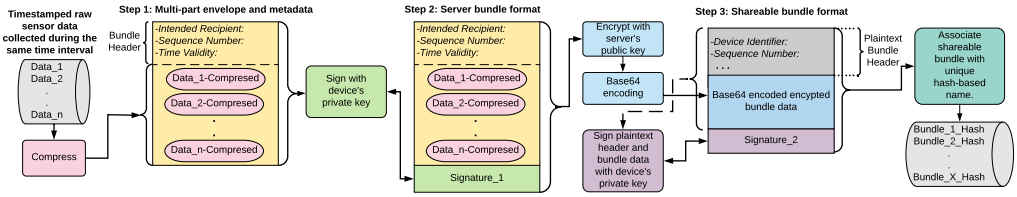


Figure 3.2: The bundle creation pipeline, read from left to right.

ing can benefit from the use of more information than just the hash/name.

The bundle structure allows sensor data to be encrypted by each node/OBU with keys that only the backend server can decrypt. However, it also includes plaintext metadata to keep track of bundles that have already been uploaded. This appeared to be a reasonable tradeoff between confidentiality of the data (since raw data can be encrypted) but reveal information, such as the sequence number which, as we will see can be used to schedule which bundles should be given priority to infect other OBUs or for upload to backend severs.

The use of additional application-layer encryption is a way to address the expanding concerns of wireless security. Specifically, the explored IEEE 802.11 P2P protocol, ad hoc - i.e. Independent Basic Service Set (IBSS), offers a baseline of rather suboptimal accessible security mechanisms for group formations – i.e., WEP. As such, and with the assumption that encounters between vehicles could be sparse and short in duration, we have decided to opt out of additional authentication schemes at the WiFi data link layer that could further delay connection establishment between vehicles, and instead endow with encryption the application endpoints.

3.2.2 Sample Data Size

Theoretically it is possible for a data acquisition program to generate data of any size. In practice, however, the size of extracted data is usually quite small compared to the size of available memory/storage hardware. The cost of commodity class portable storage options like micro SD cards is low and constantly reducing in physical size and price. As an example, we were able to extract between 50-500 bytes/second of data from each sensor. This includes full GPS output, full accelerometer output, and 15 vehicle descriptors

via OBD-II port once per second. Included in this estimate is the date and time, prepended to each individual data-point.

Figure 3.2 shows that each sensor output file, *Data_n*, is compressed to further reduce its size. The exceptionally repetitive nature of the data renders it highly compressible. With roughly 90% compression achievable, our hourly storage quota for each sensor was less than 200 KB. Looking again at step 1 of Figure 3.2, and assuming we wish to bundle all sensor data every 10 minutes, each multi-part envelope in our testing encompassed less than 80KB including the bundle header. The size will increase by roughly 33% by step 3 because of the Base64 conversion after encryption. This yields a final shareable bundle format close to 100KB per 10 minutes and 600KB per hour. Estimating within an order of magnitude we assume approximately 1 MB of stored bundle data per hour, per vehicle. This number is important for two reasons: 1) it is possible to calculate the total number of bundles delivered per day which will inform the VDTN performance analysis, and 2) it makes possible to determine the number of bundles storable on each OBU. As of the writing of this thesis, the cost of a 128 GB micro SD card is $20 CAD depending on the manufacturer, allowing for each vehicle to store at least 100,000 hours of bundled data – generated by the vehicle, or received and carried from other vehicles. This number heavily influenced our decision to use an epidemic style of routing in our proof-of-concept implementation, as opposed to some of the more memory-efficient but less "infectious" styles of opportunistic/DTN routing protocols that have been reviewed in Chapter 2.

3.3 Data Transfer Logic

Figure 3.1 indicates the four ways in which the OBU can connect wirelessly with off-vehicle endpoints. Our V2I involves connecting to known whitelisted WiFi APs for uploading data over the Internet and is dominated by the *Data Uploading* step. In terms of V2V communication we use WiFi-based P2P protocols to form ad hoc (IBSS) networks between OBUs so that data can be shared. The V2V communication is the *Data Sharing* part which opportunistically connects in P2P fashion with other OBUs to share (“infect”) bundles as quickly as possible. Each OBU tries to infect the OBUs of other vehicles

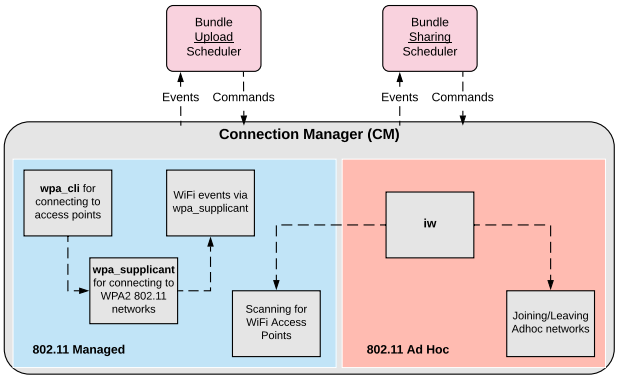


Figure 3.3: Connection Manager control flow and interaction with upload and sharing scheduler processes.

with their own bundles, increasing the chances the bundles will be delivered to backend servers, after possibly infecting several OBUs.

3.3.1 Data Uploading

Influenced by our original RPi prototype platform limitations, we use only a single integrated WiFi module for both *Data Uploading* and *Data Sharing*. Apart from that, the logic required to actually upload bundles to a server is different than what would be required to share bundles among peers. To simplify things, separating tasks based on layers of abstraction is necessary. To this extent, a Connection Manager (CM) was created to manage all wireless connection-related functionality. It handles scanning, connecting, and disconnecting from WiFi APs and P2P networks. The CM runs in the background, awaiting commands and delivering event messages to any process attached to it. A library of functions for communicating with the CM has also been created. Through this library, application layer socket programs can focus solely on uploading and sharing bundles, without concerns about lower link layer connection details. In our system, two application modules converse directly with the CM, handling all bundle scheduling decisions. Scheduling falls into two categories: *Bundle Uploading* and *Bundle Sharing*. Figure 3.3 further illustrates the interaction between schedulers and the CM.

The CM periodically and consistently scans for known WiFi networks, sending messages to the bundle schedulers whenever a network is available or unavailable. The *Bundle Upload Scheduler* in turn uses these availability

messages to initiate connections with known APs. When connected to an AP, a TCP socket connection is established with the backend server, followed by an advertisement of all owned-bundles with which the OBU wishes to upload. The server checks its database of previously uploaded bundles, and informs the vehicle, so as to avoid duplicate uploads. The vehicle sends the needed bundles to the server, which records the time of each bundle as it is received and updates its database appropriately, sending Acknowledgements (ACKs) back to the vehicle. Once uploading is finished, they both synchronize their databases with each other. The information in this database indicates the list of acknowledged successfully delivered bundles to the backbone server. The OBU can then refrain from further *Bundle Sharing* distribution of the acknowledged bundles and can free up local storage. Note that this database can also be used for the distribution of public keys, including possibly additional Public Key Infrastructure (PKI) support operations, such as key revocations, etc.

3.3.2 Data Sharing

Whenever OBUs are not participating in *Data Uploading*, they continuously and opportunistically try to connect to each other in P2P fashion. The *Bundle Sharing Scheduler* is not concerned about the particular way in which P2P communication is established – this is an issue for the CM. The scheduler simply needs to know the status of WiFi connectivity, i.e., whether or not *Data Uploading* is taking place. Upon receiving notification from the CM that WiFi has been disconnected, the scheduler sends a command instructing the CM to join whichever P2P network it has employed (IBSS in our case). An application-layer socket-based protocol is used by the scheduler for achieving optimized bundle exchanges during *Data Sharing*.

In order to create a more responsive program, the scheduler supports two channels of communication between peers. These channels are implemented through separate ports for control messages and data messages. At a high level, the process can be described in the following way:

1. As an OBU joins an IBSS network, it immediately starts advertising it's own IPv6 link local address on a shared multicast group. At the

same time, it also joins this *Advertisement Group* and listens for IPv6 addresses from other OBUs. The main advantage for using IPv6 link local addresses is the ability to instantly use them without the latency of a request/response protocol such as DHCP. A caveat is that Duplicate Address Detection (DAD) must be turned off, to reduce address assignment delay.

2. When an IPv6 address is received by an OBU, it initiates two TCP connections on separate sockets for control and data communication.

3. OBUs exchange their local bundle lists over control sockets. Upon receiving these lists, they check their databases for matches and report them to all peers so they can update their own databases and refrain from sending these bundles.

4. Using the information from the previous step, OBUs decide which bundle they will transmit next, and to whom it will be transmitted. This decision is based on a prioritization scheme. The current implementation of the scheme favors *least-owned bundles* primarily, and bundle sequence numbers are used for tie-breaks (the smaller the number, the higher the priority). All prioritization decisions can be made by using information provided in the header of the shareable bundle format. *Least-owned bundles* only comes into play when a OBU is connected to more than one other peer OBU at any given moment. That is, every local bundle has a priority number attached to it, referring to the number of peers that also own that bundle.

   As an example, consider an OBU A that is connected to two other peers, B and C. OBU A owns bundles 1, 2, and 3. OBU B owns bundles 2, 3, and 4. OBU C owns bundles 3, 4, 5, and 6. From A's perspective, bundle 1 is the least-owned bundle, so it will be sent first. The choice of who to send it to, B or C, is decided by total number of bundles owned. In this case, B owns fewer bundles than C (3 v.s. 4), so A will send bundle 1 to peer B. Based on these rules, it can be inferred that each OBU can only send at most one bundle at a time, but it can receive bundles from

multiple peers at the same time. If more than one peer holds the same bundle, the peer with the higher IPv6 address will be the one to send it.

5. All bundles are transmitted on data sockets. This frees up control sockets for sending urgent messages, such as those indicating newly received bundles. Peers can either accept or reject the bundle before it is sent. This helps prevent identical bundles from being sent to a peer at the same time. Such scenarios can occur in IEEE 802.11 IBSS mode when a device "in the middle" is connected to two neighbors that cannot see each other.

Our data sharing approach necessitates a lossless communication medium. This, coupled with comparable throughput analysis for both TCP and UDP helped us choose the former as our transport layer protocol. Ideally, in situations where large groups of vehicles are connected together, a broadcast or multicast approach would seem more appropriate than unicast for a data sharing algorithm that aims to disseminate data to as many peer devices as possible. Unfortunately, our testing with all three WiFi based connection standards: infrastructure (AP) mode, ad hoc (IBSS) mode, and WiFi Direct, yielded unacceptable multicasting throughput of approximately 1 Mbps. Comparing this to unicast, which peaked at 30-40 Mbps at close distances, it is obvious why a unicast approach was chosen.

3.4 Implementation Lessons

Creating a set of programs and libraries that interface to work with existing free/open source software presents itself with several challenges. On top of that, everything must be automated to work on Linux systems.

`wpa_supplicant` and `iw` vs. `netlink` socket

The decision to use `wpa_supplicant` was made early on for several reasons: (1) It allows devices to connect to WiFi networks that incorporate security protocols like WiFi Protected Access 2 (WPA2). (2) It is capable of both receiving requests as well as streaming event messages. (3) It is possible to link with

some of its object files, allowing our CM to connect via domain socket and receive event messages continuously. This last point is very important because it allows our CM to monitor event messages directly from `wpa_supplicant`. With this information the CM can easily keep track of the current state of the network (connected to WiFi AP, P2P IBSS, or neither) and provide meaningful information to our bundle scheduler programs (WiFi and IBSS network availability) so they know when to attempt their respective socket connections.

The Linux program `iw` is a wireless configuration utility capable of scanning and handling basic network connections. `iw` is lightweight and easy to use. Under the hood it uses `netlink` sockets to communicate with the network device drivers. Ideally, `iw` would be used by the CM for all wireless connection functionality - both WiFi and IBSS. In practice, however, it lacks the ability to handle security protocols like WPA2, and it is not capable of directly providing event messages based on network status. In our testing code-base, `iw` is only used for managing our P2P connections via 802.11 IBSS networks.

An alternative approach would be to bypass `iw` and write code that directly uses `netlink` sockets. As opposed to `ioctl` system calls, `netlink` sockets are more flexible and capable of listening and receiving messages in similar fashion to domain sockets. Unfortunately, `netlink` socket programming is quite esoteric, so it is not as easy to write code using it. Ultimately, our CM makes system calls to execute commands via `iw`.

Example system flow of utilities (assuming an initial state where the OBU is not connected to any network):

1. CM joins the IBSS network and continuously scans on regular intervals for available APs via `iw`.

2. *Bundle Upload Scheduler* blocks until a known AP is found.

3. *Bundle Sharing Scheduler* grabs its link local IPv6 address and broadcasts it on the P2P IBSS network while at the same time listening for broadcasts from other OBUs.

4. When an OBU hears a broadcasted IPv6 address from a peer, it attempts to initiate a socket connection with the peer and initiate bundle exchange.

5. At any moment in time, as soon as a known AP is found via scans from step 1, the CM disconnects from the IBSS network and switches the network card to act as a managed WiFi station via `iw` to connect to the found AP.

6. `wpa_supplicant` is then started. It uses a configuration file to recognize and connect to the AP.

7. Once a connection is established, the CM is notified by `wpa_supplicant`, which in turn notifies the upload scheduler.

8. The *Bundle Upload Scheduler* then initiates socket connection over the internet via the AP to a backend server which receives all bundles for the entire system.

9. Upon disconnection from the AP we go back to step 1.

The entire process is a bit sluggish and involved, because:

One vs. two WiFi network interfaces

One of the constraints of our prototypes was the use of a single integrated network card for all network activity. The addition of an extra network adapter would drastically simplify the CM. With two network cards, one dedicated to WiFi connections and the other dedicated to P2P communication, it would be possible to:

- Completely negate the need to preemptively scan for WiFi networks. This would also improve throughput performance because scanning is expensive in terms of network card usage and, ultimately, slows down bundle transfers with other vehicles/devices.

- Stay connected to an IBSS network all the time, simplifying CM logic. It will only need to report connect/disconnect messages from `wpa_supplicant` to the *Bundle Upload Scheduler*.

- Leave `wpa_supplicant` running all the time, allowing for more immediate WiFi connections when available.

Regardless of quantity of network cards, another important consideration - for P2P IBSS *bundle sharing* specifically - is the choice between:

TCP vs. UDP

In theory, UDP yields higher throughput than TCP due to its connectionless nature, which is further confirmed by the results in Chapter 7. In practice, our throughput tests between stationary devices showed virtually similar rates between the two protocols. With this assumption, it would seem like TCP is the better choice because all re-transmissions of packets are handled at lower layers in the networking stack as opposed to UDP that requires lost datagrams to be handled at the highest level of the stack: i.e., the *Bundle Sharing Scheduler*. Using this logic, a choice was made to implement TCP connections - not UDP - for all P2P IBSS transmissions within OBUs.

Ideally, using TCP would negate the need for retransmission logic at seemingly no cost. There is a cost, however, and it has to do with the transient nature of our wireless networks. If one were to assume, in opposition to our VDTN, a reliable wireless network like a Wireless Local Area Network (WLAN), making a Linux *connect* system call would be relatively straight forward. With a VDTN, however, encounters between vehicles are not only infrequent, but not always binary in their connected nature, as described in the related work in Chapter 2. As vehicles approach each other, it is most likely they will initially "hear" each other at the far edges of their transmission range. Not only is the network unreliable at this distance, but it is also not guaranteed. This can overly complicate the programming logic and ultimately effect efficiency of general uptime.

To illustrate this point, we consider an example encounter scenario between two vehicles. Imagine vehicle A and vehicle B are both traveling in a similar direction. A is behind B and their traveling speed is very similar with slight fluctuations. They are close enough physically that occasionally vehicle A hears a broadcast message from vehicle B advertising its own IPv6 address. Hearing this address, vehicle A would then initiate a TCP *connect* operation with B. Here is where the difficulty arises. If vehicle B does not receive the TCP handshake request caused by the *connect* call, or if it does but vehicle A

does not receive the response, then vehicle A may have to proactively suspend the connection attempt. While it is possible to set up non-blocking *connect* system calls on Linux - effectively bypassing the inevitable lengthy wait on a blocking call - the code becomes unnecessarily complicated.

Continuing with the previous example of vehicles A and B moving in tandem with varying speed in such a way that they continuously move in and out of transmission range of each other, bundle transfers themselves also become increasingly more difficult. Manual timeouts must be implemented and managed in order to avoid hanging sockets. This further complicates the codebase. Moreover, our bundles are not fragmented in any way during transmission, so incomplete transmission of a bundle means it must be re-sent from scratch in a future opportunity. This simplifies transmission logic, and, due to the dissemination style of our routing protocol, coupled with the ability to control the size of bundles, we consider it is a reasonable tradeoff.

Chapter 4

The Simulation Platform

The purpose of simulating our VDTN scenario is to study at-scale the platform developed, even if abstracted to some degree. To do so, we make use of actual urban road maps and practical mobility patterns that emulate a realistic VDTN inspired by business needs for managing and tracking company-based fleet vehicles. The simulation itself can be generalized to a certain degree, but the following were considered as the key aspects to capture, and considerable effort was invested to this effect:

1. ability to accurately simulate the movement of a small number (in the 10s of vehicles) that comprise a company's fleet,

2. ability to accurately simulate the movement of a significant volume of other, non-fleet, vehicles that compound road congestion and create more realistic traffic conditions,

3. separation of the simulation into intervals representing days, able to reflect the working hours during which the vehicles are used each day,

4. inclusion of at least one depot from which vehicles depart at the beginning and return to at the end of working hours,

5. inclusion of a coarse scheduling logic at a fairly large grain, assigning each fleet vehicle to several stops within a sector of the map,

6. inclusion of non-work stops, capturing opportunistic visits of vehicles to a set of predetermined “rest stop" locations,

7. inclusion of non-work stops, past which vehicles may travel but are not "rest stops", to capture the locations of co-located WSN gateways,

8. ability to record location data throughout the simulation, to produce the necessary encounter statistics,

9. ability to separate recorded location such that the analyst can track encounters between:

   (a) any two or more company fleet vehicles,

   (b) any fleet vehicles and stationary rest stops, and

   (c) any fleet vehicles and stationary WSN gateways.

The inclusion of the rest stops is subsequently necessary to model transfers of data from vehicles via the rest stop WiFi APs which are typically publicly accessible. The inclusion of WSN gateways is needed to capture the case of data generation that is encounter/event-based, i.e., “WSN data mule” role, rather than proportional to the travel distance, as explained in Section 1.5. It is worth noting that while the drivers are aware of the rest stops, there is no need that they are aware of the location of the WSN gateways. It is important that the simulator captures all complexities that exist within the topology of real city maps, for example, the varying speed limits and traffic rules that exist within these cities such that the encounter statistics are realistic. An important distinction of the work in this thesis compared to most previous works, is that the encounters between two devices are analyzed at various distances. More details to this effect are provided in Section 5.3. In the remainder of this chapter we emphasize on the vehicular simulation aspect, moving the emphasis on the encounter statistics in the next chapter.

4.1 SUMO: A Vehicular Mobility Simulator

As noted earlier in the related work chapter, a well-known vehicular mobility simulator, that we adopt for our work, is SUMO [13]. The degree of realism it attains is high because it is able to import road maps from real cities, and simulate pragmatic vehicular traffic movement. The strengths of SUMO can be summarized as follows:

1. road networks can be imported from real city maps using, e.g., open source city maps acquired from `www.openstreetmap.org`,

2. actual traffic rules (signals, signs, multiple lanes, lane changing, traffic lights, yielding, and merging) are supported,

3. vehicles' speed behavior is realistic and specific to roads, including acceleration and deceleration as vehicles approaching and leaving,

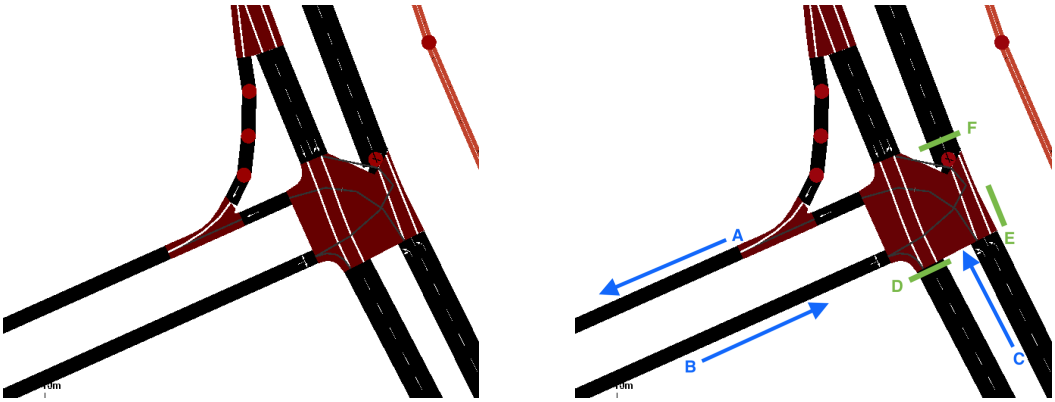
4. several libraries exist for connecting and controlling SUMO during simulation, used also for storing a record of the location, velocity, lane position, travelling direction, etc. at each time step.

4.1.1 Limitations of SUMO

There are, however, some resultant behaviors from SUMO that have the potential to cause anomalies which may stray slightly from realistic traffic movement. Every effort was made that they did not occur, or if they occurred, that their impact to the simulation was minimal.

- Traffic Light Timing: These can be non-representative of actual traffic needs (requiring manual intervention to configure properly). An example would be a standard cross intersection that experiences heavier through-traffic on one of its roads as opposed to the others. In a real city, heuristics are used to manipulate traffic light timings in ways that prioritize these types of intersections. Another example would be turn-lanes that undergo heavy utilization.

- Intersection Geometry: The conversion program provided by SUMO, named `netconvert`, is not always able to ascertain exact intersection geometry. As such, many real-world intersections are not accurately represented within SUMO, and sometimes this can result in vehicles getting "stuck", causing an impasse. In these cases, manual intervention is needed to see if the general behavior appears correct. There are many instances where the number of traffic lights and turn lanes differ greatly from satellite imagery. Figure 4.1 shows an example intersection converted from `www.openstreetmap.org` that differs considerably



(a) Intersection representation after converting with `netconvert`.

(b) Intersection representation informed by satellite imaging (Google maps).

Figure 4.1: Intersection geometry, (a) based on `www.openstreetmap.org` data via `netconvert`, (b) with added missing lanes inferred from Google maps data. Missing lanes are indicated with labels A, B, and C. Missing traffic lights are shown with D, E, and F.

from reality. The blue lines represent missing lanes, and the green ones highlight absent traffic lights. Rather than search for and fix all such misaligned intersections, we have chosen to let traffic congestion dictate which intersections to apply scrutiny towards. It is not as important to mirror exactly the city that is being modeled as it is to at least represent topology that, at the very least, works on a realistic and pragmatic level.

- Vehicle Teleportation: When the velocity of a vehicle is below 0.1 m/s (essentially stopped) for at least a set period of time, it is removed from the network and advanced along its intended path until an opening is present, at which point it is injected back into the network. This is a feature that prevents infinite deadlock due to traffic jams caused by imperfect traffic-light rules, high traffic flow, or imperfect intersection geometry. Teleporting is often the result of the previous two issues. These behaviours are further described in the SUMO wiki [14]. We keep track of teleportation frequency, and present this information when we analyze the simulation results. Its impact was found to be minimal.

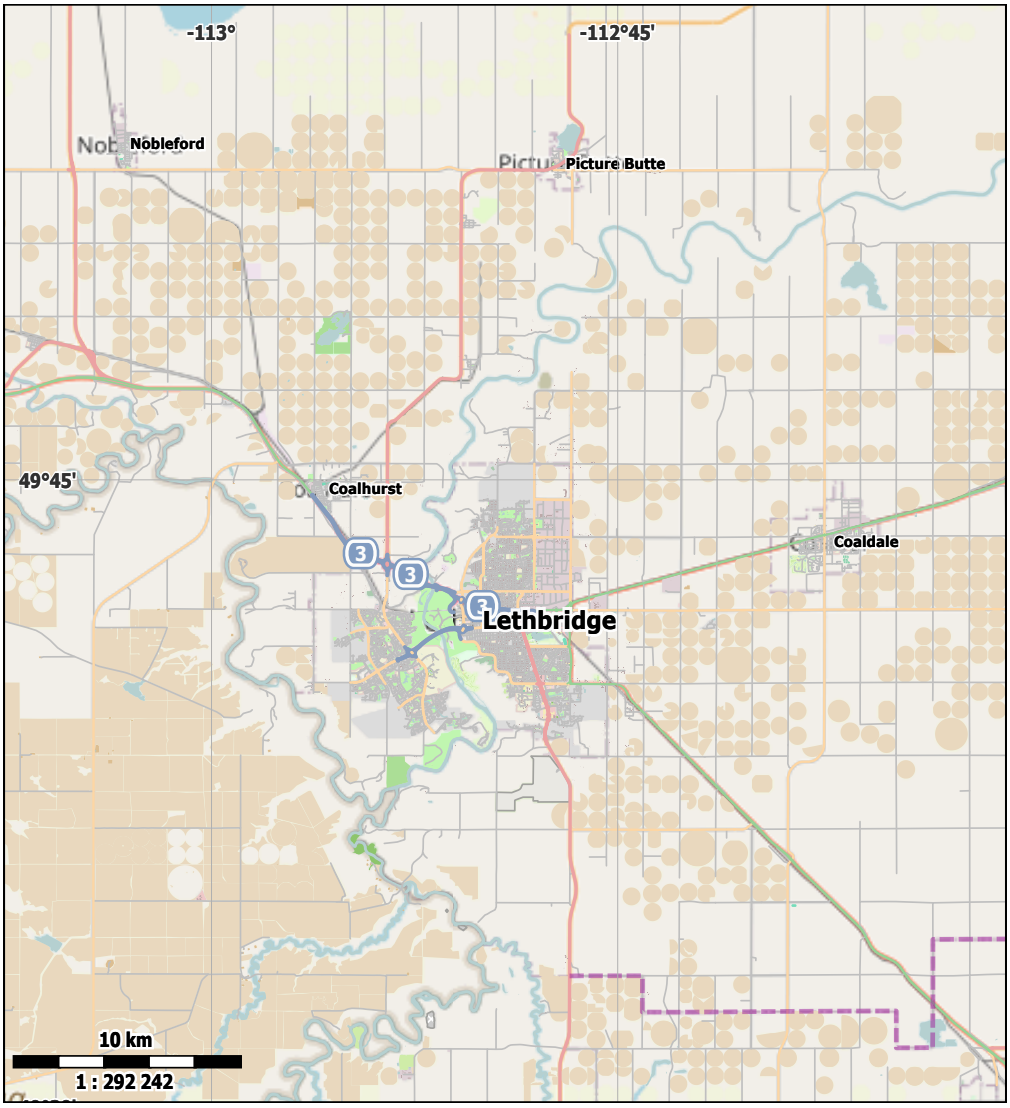


Figure 4.2: Lethbridge and surrounding area from `www.openstreetmap.org`

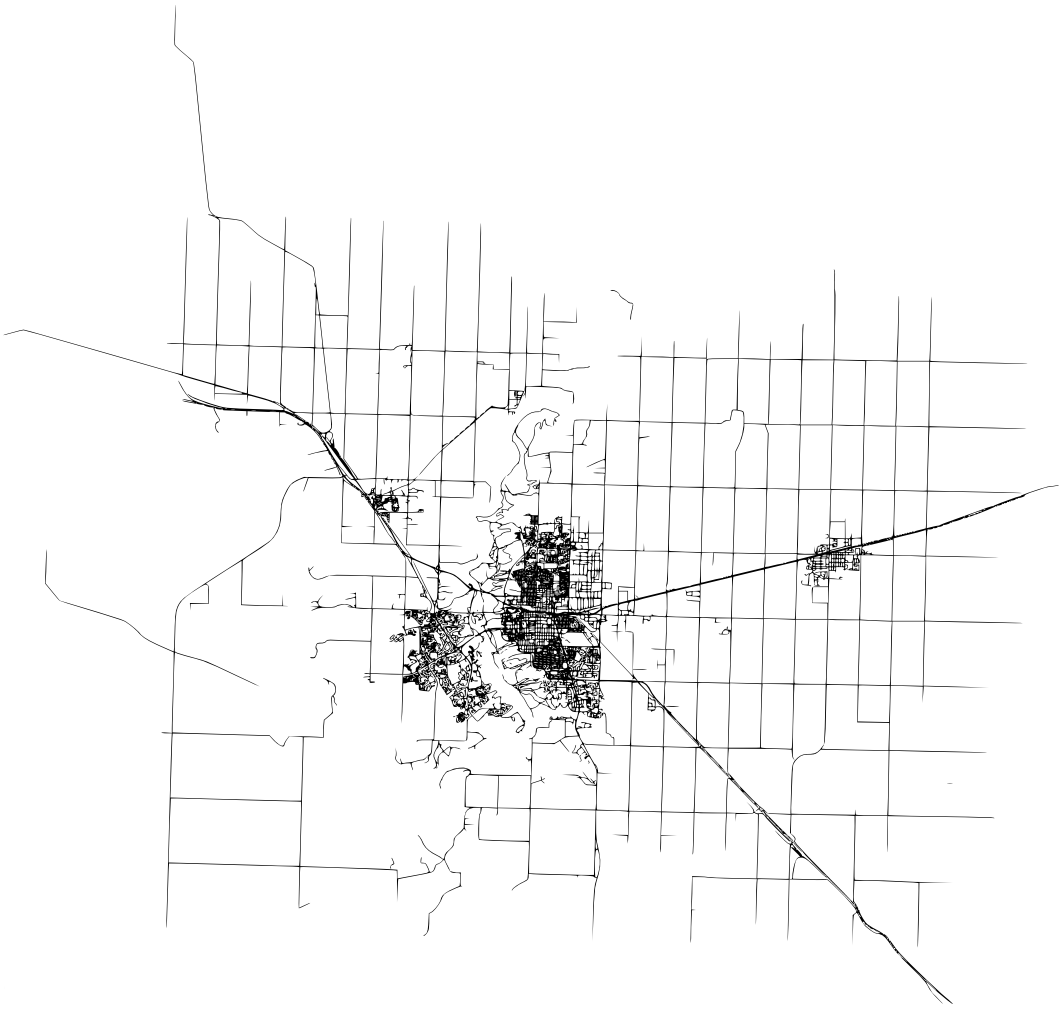


Figure 4.3: Road network of, and around, Lethbridge used in the simulations. Notice that the adjacent communities of Coalhurst and Coaldale are included.

4.1.2 Road Network Layout Pre-Processing

SUMO road network maps are Extensible Markup Language (XML)-encoded files describing, essentially, directed graphs, where the edges of the graph form road segments, and the nodes represent connections between edges. Some of the connections describe road intersections, otherwise known as *junctions* in SUMO terminology, while others are used to connect multiple edges together, forming what would be seen as a continuous street in a real-world setting. For example, a road connecting two city blocks could be encoded as a single SUMO edge, multiple edges daisy chained together to form a whole, or even a small segment of a much larger edge that expands past the two city blocks. Edges may also contain multiple lanes, and must be unidirectional. Apart from these basic elements, SUMO also supports configurable traffic light rules for intersections.

There are two generally accepted ways of creating a network map for SUMO to work with. One is to start from scratch (either manually editing xml files, or by using an accompanying Graphical User Interface (GUI) program called `netedit`); the other is to import an existing road network from a supported input source using a program called `netconvert`. One of these supported input formats is the OSM file format, which can be gathered from `www.openstreetmap.org`, which provides a free map of the world with options to save user-controlled map segments. Figure 4.3 shows a resulting conversion of a map segment from `www.openstreetmap.org` to SUMO network file format using `netconvert`. SUMO maps have the extension `.net` and are referred to as *network* maps.

For the purposes of the simulation work conducted in this thesis, we have used the `netconvert` approach of `www.openstreetmap.org` maps, as it is the only automated option. It is worth mentioning some options that are available to `netconvert` that were used in processing the maps in our simulations, and their significance, quoted from [12]:

```
''
--geometry.remove: Simplifies the network (saving space) without
 ↪  changing topology.
```

```
--roundabouts.guess: Sets the appropriate right-of-way rules at
 ↪  roundabouts. (Explicit right-of-way rules are not imported
 ↪  from OSM). If this option is not used and roundabouts are
 ↪  not defined manually, then traffic jams will likely occur at
 ↪  roundabouts
--ramps.guess: Best guess addition of acceleration/deceleration
 ↪  lanes. Often acceleration/deceleration lanes are not
 ↪  included in OSM data. This option identifies likely roads
 ↪  that have these additional lanes and causes them to be
 ↪  added.
--junctions.join: Joins junctions that are close to each other
 ↪  (recommended for OSM import).
--tls.guess-signals: Interprets tls nodes surrounding an
 ↪  intersection as signal positions for a larger TLS. This is
 ↪  typical pattern for OSM-derived networks
--tls.discard-simple: Does not instatiate traffic lights at
 ↪  geometry-like nodes loaded from other formats than
 ↪  plain-XML.
--tls.join: Tries to cluster tls-controlled nodes.
--no-turnarounds: Disables building turnarounds.
--no-turnarounds.except-deadend: Disables building turnarounds
 ↪  except at dead end junctions.
''
```

Finally, `netconvert` also categorizes the edges that make up roads based on the types of vehicles (including pedestrians) that are allowed access on them.

4.2 Vehicle Routing

We consider the behavior of *fleet* vehicles to be separate and defined compared to the behavior of "background" *traffic* vehicles. This is a modeling contribution necessary to separate the behaviors of interest. For clarity's sake, we define *fleet* vehicles to be those which belong to the VDTN we are interested in tracking. They are considered to belong to a company whose interests involve monitoring vehicular activity.

The goal of *traffic* vehicles is to maintain as much as possible a consistent average number of vehicles in the map during the simulated *business hours*. Here, we define *business hours* to be a rigid interval $[B_{start}, B_{end}]$, where B_{start} and B_{end} refer to the beginning and ending of the work day, in seconds, using standard 24-hour clock, respectively. These hours may also be referred

to as *working hours*. *Traffic* only needs to be maintained during business hours, as that is the frame with which we wish to analyze the *fleet* vehicles. In principle one can model the entire 24 hour period, but it has no material impact in our examples.

4.2.1 Background Traffic Behavior

In terms of routing these anonymous *traffic* vehicles, source and destination locations are randomly chosen from all available network edges. Travel starts at the beginning of the source edge and stops at the ending of the destination edge, where the vehicle is then removed from the network. Routes are chosen traffic-oblivious for shortest travel time based on speed limit and road distance. For reasons of simulation run-time efficiency, routes are randomly chosen from a (large) pre-calculated set, relieving the simulator from performing the same, or similar, path calculations during its execution.

To ensure a relatively fixed population within the boundaries of the map, no vehicle trajectories are defined to exit the limits or enter the space of the map we are simulating. This is an artificial constraint but necessary to provide direct control of the population of *traffic* vehicles.

We define the number of *traffic* vehicles in the road network during business hours as a fluctuating average falling within the interval $[N_t - N_d,\ N_t + N_d]$ vehicles, where N_t is the desired number of traffic vehicles, and N_d represents a delta value used as a threshold for establishing lower and upper bounds. When the current number of traffic vehicles being simulated, N_c, falls below $N_t - N_d$, a fixed number of vehicles are added to the network during each subsequent simulation time step until $N_c \geq N_t + N_d$. The rationality for maintaining a traffic threshold is purely pragmatic. We do so to improve simulation performance (through alleviating the need to check current traffic numbers during each simulation step).

The precise rule-set used for managing traffic vehicles during simulation is as follows:

1. The traffic threshold is only maintained (traffic vehicles are only added to the network map) during the interval $[B_{start} - 1200sec,\ B_{end}]$. The reason for introducing traffic vehicles into the simulation 1200 seconds

before the start of the work day, is to allow for a gradual increase of vehicle density. This "ramp up" period helps to avoid possible congestion in certain areas as vehicles are flooded onto the map. It also allows time for a more realistic geographical distribution of vehicles to be attained by the start of the work day.

2. Every 60 seconds, during *working hours*, the total number of vehicles currently in the network is acquired and compared against the set lower bound of the traffic threshold interval, i.e., $N_t - N_d$. If it has fallen below this lower bound, vehicles are added to the network each simulation step until $N_c \geq N_t + N_d$.

3. Each added vehicle is given a route randomly chosen from a list constructed prior to simulation, as explained earlier. Additionally, to increase the length of time each vehicles spends on the road, and create a more diverse distribution of vehicles, each of these routes has been given five intermediate waypoints. Consequently, by increasing the driving time of traffic vehicles, we reduce the number of vehicles added to the network during the course of each work day, while still allowing them to be removed from the network gracefully by SUMO as they reach their destination edges.

4.2.2 Fleet Vehicle Behavior

We define the number of *fleet* vehicles being simulated as a constant value N_f. We monitor their behavior and record statistics about each vehicle during every simulation time step. For all intents and purposes, these vehicles represent the complete set of vehicles belonging to the VDTN we are modelling. We imagine a company with a fleet of vehicles which (a) leave a depot at the beginning of a work day, (b) perform several service stops at customer establishments, spending a specified amount of time at each stop, (c) may take breaks at particular times at rest stops that contain WiFi APs (d) return to the closest depot at the end of the work day. It should be noted that the city map is divided into geographical *sections* representing a coarse form of spatial scheduling we consider reasonable for a corporate fleet. That is, each fleet

vehicle is assigned a *sector* to operate within, allowing for a relatively even spread of vehicles across the entire map. In our simulations the number of sectors is four.

All *fleet* vehicles operate within the confines of the working hours that have been previously defined. Rather than having all *fleet* vehicles leave their respective depots at the exact moment the work day begins, we take the more realistic approach to define a period of time following the start of the work day with which *fleet* vehicles are permitted to embark. Using this model, each day, vehicles are randomly assigned a starting time within this period.

4.2.3 Fleet Routing Behavior

Fleet vehicle routing differs considerably from traffic routing. During the simulation of the working hours, our script gathers various information about all *fleet* vehicles at each simulation time step, and decides the correct course of action. The exact steps are described in detail below. We have separated the logic into three categories that dictate the way in which vehicles operate.

- **Depot Departure:**

    1. At the beginning of each work day, all *fleet* vehicles are given a departure time. This represents the exact time they leave their respective depots. These times are chosen uniformly randomly on the interval $[B_{start}, B_{start} + T_{sp}]$, where T_{sp} represents the length of time in seconds past the official start of the work day with which *fleet* vehicles are permitted to leave.
    2. As each vehicle leaves their depot, they are assigned an initial customer service stop. The stop is chosen randomly from all available road edges belonging to the geographical sector of the map to which the vehicle has been assigned. To help avoid deadlocks in traffic, edges, i.e., road segments, must meet a minimum physical length before being chosen as a place for *fleet* vehicles to stop. This avoids infinitesimally small internal edges from being considered as candidate edges. Without this caveat, vehicles could be assigned to stop in the middle of intersections, which could prove problematic.

3. A route is generated from the depot to this first stop. The *fleet* vehicle is assigned the route and leaves the depot during the next simulations step.

- **During Work Hours:** This section describes the decisions and actions of *fleet* vehicles after they have left their depots, but before returning back to them at the end of the work day. During this time, drivers operating fleet vehicles are assumed to be performing their work-related duties. These duties include driving to customer locations for service stops, and taking occasional opportunistic breaks at known rest service locations with WiFi access. It should be noted that for all of our simulations we have chosen to use exactly one depot located centrally. The service stop locations are chosen randomly from any reachable road section on the map. The specific decisions that each vehicle makes during these business hours are as follows:

  1. If a vehicle is en route to a customer service stop, it will always remain on course until arriving at its destination.
  2. Upon arriving at a customer service stop, the vehicle will pull off of the road, and remain stationary for a predefined amount of time. We refer to this stationary length of time as T_{st} seconds.
  3. When a vehicle finishes waiting at a stop (servicing a customer), it takes one of two actions:
     (a) If the current time of day is close to the end of the work day, a new route will be generated from the current service stop back to the nearest depot. To enforce this, we create a cutoff time to determine if a vehicle falls into this category. This cutoff time is defined as $T_{rt} = B_{end} - T_r$, where T_r is a length of time in seconds. It is a heuristic influenced largely by the length of time vehicles spend at service stops. The definition of T_{rt} implies that some vehicles will return before, and some after, the end of the work day, but at the very least no fleet vehicle will attempt to make another stop if their previous stop was completed after the end of the work day. On average they will

return to their depots at the end of the work day. We are not concerned enforcing these restrictions too much, but interested in creating a realistic heuristic that allows for vehicles to return home in a manner that resembles as much as possible a random distribution with acceptable lower and upper bounds.

(b) Otherwise, the vehicle will be randomly assigned a new service stop location in its sector. A new route is generated for the vehicle heading towards this new stop.

Example: Assuming customer service stops are 20 minutes in length, a fleet vehicle finishes a stop at 4:41pm and the end of the work day is 5:00pm, it will then determine there is not enough time to make another stop, and return to the depot, possibly arriving earlier than the end of the work day. If, however, the vehicle finishes a stop at 4:39pm, it will attempt to make another customer stop before going back to the depot. These examples illustrate the variance in depot return times for fleet vehicles.

4. Lastly, to accommodate the need for employees to take breaks, as well as allow for access to public WiFi APs, fleet vehicles are required to make quick stops at any of the predefined rest areas *if* they happen to drive past these areas during their current route. The duration of these stops is a (very conservative) interval of two minutes.

- **Depot Return:**

1. Upon reaching their respective depots, *fleet* vehicles remain at the parking area until the start of the next work day, when they are given a new randomly assigned start time.

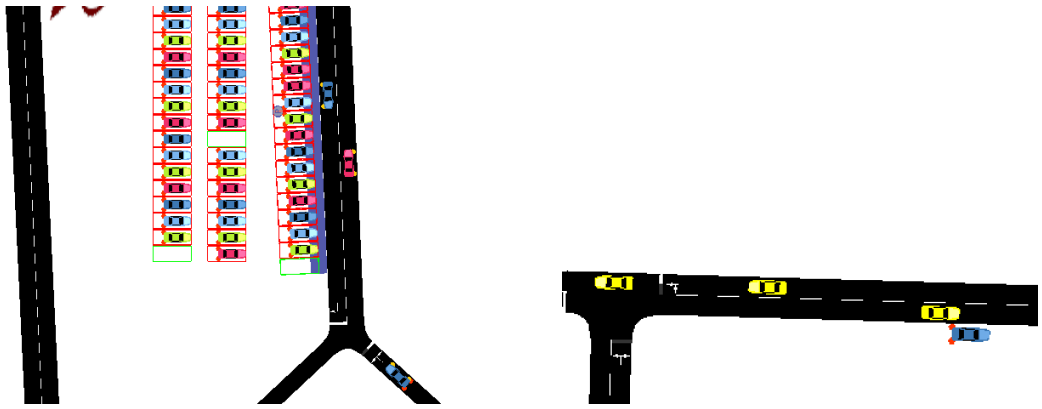
Finally, note that the map area, which is used to simulate vehicles by SUMO, represents a closed system, as can be seen in Figure 4.3 and there are no trajectories going outside or coming into the map as no such points of origin/destination are possible to describe. This means that any roads leaving or entering the enclosed rectangular area are considered to be dead-ends. This

affects routing, but it is unavoidable without considerably altering the map, or simulating the road network of an entire continent! As such, fleet vehicles are only assigned routes with which have loop-back (return) potential.

4.2.4 Areas of Interest

There are two types of areas of interest. In the first category are the ones with which we can associate a parking area. The second are those where the fleet vehicle drives by without stopping, i.e., the WSN gateway locations. For the first category, there is a need to model parking.

Parking Details



(a) Parking area with three empty stalls outlined in green.

(b) Roadside parking stop with fleet vehicle (blue) and traffic vehicles (yellow).

Figure 4.4: Parking area (lot) vs. road-side parking.

SUMO provides different methods for simulating vehicular parking scenarios. We differentiate between two important methods here. Figure 4.4(a) illustrates what SUMO refers to as a *Parking Area*, and Figure 4.4(b) shows an example of what it means for a vehicle to simply stop on the side of the road, showing how the *fleet* vehicle does not impede other vehicles on the road. For the latter of these two examples, the blue vehicle is both stopping and parking, allowing it to be removed from the road, and other vehicles to pass by. This type of parking is handled entirely by SUMO during runtime and requires no manual intervention regarding manipulating the network structure (road map). *Parking areas*, however, must be created prior to runtime through

additional network files that describe the location, size and number of stalls, and rotational orientation of these areas. We can see the manually addition of a parking area in Figure 4.4(a) between the two vertical roads. This parking area does not physically exist in the actual city map we are simulating.

For both of these parking methods, SUMO does not simulate all of the micro details entailed by parking. Rather, vehicles slow down to zero velocity at the closest position to there desired parking location and then instantly reposition themselves. Using Figure 4.4(a) as an example for parking areas, the blue vehicle on the black road traveling southbound will gradually come to a stop and then immediately be transported to one of the available parking stalls. In Figure 4.4(b) the blue vehicle was moving eastbound before slowly coming to a stop immediately above its shown parking location. It was then re-positioned adjacent to its final location. The repositioning happens in a single simulation step.

In our simulation scenario, regular traffic is never preemptively stopped. *Fleet* vehicles, however, will stop under three situations:

1. *Fleet* depots make use of *parking areas* to emulate a realistic parking environment that can handle a considerably high number of vehicles in a manner that makes efficient use of space and provides realistic proximity between vehicles.

2. When *fleet* vehicles stop for breaks at predetermined rest stops locations, those locations also employ *parking areas*. This allows more than one vehicle to be parked without having to worry about running out of space. It also allows for controlled positioning that promotes realistic proximity to the rest stop establishment, and hence to its WiFi infrastructure.

3. Roadside parking stops (Figure 4.4(b)) are used when *fleet* vehicles make customer service stops. This allows for them to stop at any network edge (road segment) as needed, without having to manually add a parking area beforehand.

In summary the inclusion of parking stops does not involve any manual configuration, but the same is not true of parking areas. Nevertheless one can uti-

lize user-generated templates of parking area patterns and reuse them across simulations.

WSN Gateways

WSN gateways are areas of interest that are manually placed throughout the map. They are represented and referenced internally by SUMO as Points of Interest (POI). They do not change the routing behavior of any vehicles. Their placement is unrestricted, so one can, in principle place them a significant distance away from road segments – although the wisdom of doing so for VDTN purposes is debatable, and hence all gateways in our case are placed next to a corresponding road segment. The assumption is that the gateway connects to a complete WSN, possibly utilizing wireless multi-hop routing separately and independently from the VDTN. In our current simulation, we do not make any attempt to model the amount of traffic generated by the WSN that needs to be delivered through the gateway to fleet vehicles driving by, but doing so is a straightforward extension.

Chapter 5

Simulation Results

Among the key observations highlighted in the related work (Sections 2.1.1 and 2.2) was that, as vehicles approach each other, the bit rate adaptation results in the communication speed continuously changing, depending on the distance at any instant during that encounter. Assuming a threshold distance, D_{max}, then an encounter can be defined as the period from two nodes coming closer than D_{max} up to when their distance increases again to more than D_{max}. An educated guess for D_{max}, without loss of realism, is 200 meters because, in many studies, WiFi communication beyond 200 meters is not likely or capable to be consistently sustained, even at low bit rates. During the encounter, the nodes can get closer to, or further away, over time, until again they are more than D_{max} apart, which is the event defining the end of the encounter. The distance dynamics during an encounter correspond to bit rate dynamics of the encounter. Hence, our simulation ought to be able to track the distances during the encounter. As described in the previous chapter, SUMO will produce the locations of the vehicles at each time step of the simulation. Therefore those distances can be calculated. We expect the encounter times in an urban environment to be frequent enough so that the fleet vehicle nodes can exchange data. Furthermore, the frequency of encounters with rest stop APs provides opportunities for the vehicular nodes to upload data to backend servers throughout the working day. We also expect that the more the fleet nodes, the more frequent the encounters for data exchanges among nodes.

5.1 From Trajectories to Data Transfers

We distinguish the following possibilities for communication encounters:

1. vehicle to vehicle encounter mainly while both mobile (except if stationary because a vehicle can be parked serving a customer),

2. vehicle to rest stop WiFi, where often the vehicle is stationary (parked) and rarely as the vehicle drives by a rest stop,

3. vehicle to WSN gateway, where the vehicle is often mobile and rarely stationary (parked nearby but within range of the WSN gateway), and,

4. vehicle to depot WiFi infrastructure, where vehicles are primarily stationary as they park upon return to the depot.

The last type of communication does not reveal anything important about the VDTN data carrying capacity, as all traffic on the vehicles can be uploaded via the depot WiFi infrastructure at the end of the day. The rest stop WiFi encounters (partially) relieve the need to wait until the end of the day to upload data at the depot. The most important to track is the first type of encounters although all types are simulated and can be analyzed. This is because the encounters among vehicular nodes give a sense of the magnitude of data transfers that can take place during the encounters, averaged over e.g., the working day. As noted earlier (Section 2.6.3), we do not deal specifically with the VDTN data routing as there has been significant research work carried out in the area. Rather, we would like to express, ideally with a single number, the additional data carrying transmission opportunities possible by the introduction of the VDTN.

5.1.1 Concentric Rings Distance Model

To help in the analysis of the attainable bit rates, one way is to have a model that maps the continuum of distances to bit rates, and hence to throughput. However, our throughput values are a result of field measurements shown in the next section. Measuring throughput at any conceivable distance is practically untenable, and in any event, we are looking for informed approximations of the throughput at ranges of distances as they may differ in other

environments. For this reason, we partition the interval from a distance of zero meters to D_{max} =200 meters into sub-intervals and we collect only one throughput measurement (at the midpoint) of each interval. Informed previous WiFi performance, the following intervals (in meters) are considered: (0,10], (10,20], (20,35], (35,50], (50,75], (75,100], (100,150], (150,200], and a generic (200,∞] representing distances exceeding D_{max}. The simulation can trivially determine when thresholds between intervals are crossed, essentially treating the distances during the encounter as a set of concentric rings at distances that are the interval boundaries (10, 20, 35, 50, 75, 100, 150, and 200 meters).

5.1.2 A Markovian Encounter Distance Model

The purpose of the SUMO simulation is to assist us in building a model that describes the dynamics of the distances between vehicles during an encounter. This model will then be informed by throughput measurements to produce estimates of expected VDTN data carrying capacity. Assume two nodes who were far from each other, have just gotten less than 200 meters apart, but are more than 150 meters apart, their distance can be considered as belonging to the (150,200] interval. While it is possible their distance will remain in this interval for some time, eventually their distances may increase beyond 200 meters (hence the end of the encounter) or get closer than 150 meters. This can be generalized across all intervals, as from any interval, the two nodes may find their distance in the next time step to be within one of the two adjacent distance interval – either the one placing them further from each other, or closer to each other. From the simulations we can derive the conditional probability p_{ij} describing whether, if the distance of two nodes is within the interval i, it will be in interval j in the next time step. Note that p_{ii} is non-zero, i.e., the distance between two nodes can, with certain probability, remain within the same interval in the next time step. The intervals are modelled as states of a (discrete time) Markov chain (DTMC), and the p_{ij} are the transition probabilities between states. The steady state π of the DTMC can thus be determined, where π_i is the steady state probability that the distance between two nodes falls in the i–th interval. Later in this theses,

informed by throughput measurements, given throughput estimates b_i, the overall data carrying capacity due to the encounters can be approximated by $C_{VDTN} = \sum_i \pi_i b_i$.

In the rest of this chapter we describe the parameters, settings, and analysis carried out to reach the point where the encounter distance DTMC can be built. Subsequent chapters detail how the b_i were obtained through field measurements for various protocols and what is the resulting C_{VDTN}.

5.2 Simulation Parameters

The concentric distances pertain to the post-processing analysis of the simulation results. The parameters controlling the simulations are:

- SUMO Parameters
  - *Simulated Time* was set to ten working days (two five-day workweeks). It is adequately long for numerous encounters to happen even in the smallest fleet scenarios.
  - *Step Length* is the time between each SUMO simulation step. Reducing this value results in finer granularity of vehicle positioning at the cost of increased simulation time. As such, we have chosen to use a step length of 0.1 seconds. This value allows for reasonable overall simulation run-time given the commodity class hardware we had access to. The maximum speed limit on our city map is 100 km/h which is 27.8 m/s. This implies a vehicle can move at most 2.78 meters per time step.
- VDTN Parameters
  - *Number of Fleet Vehicles*: Throughout our simulations we explore varying number of fleet vehicles to better observe the way in which growth patterns affect encounter statistics. The chosen numbers of fleet vehicles being simulated represents a realistic amount given the size of the city they are operating within. One additional constraint is that the number of fleet vehicles must be divisible by four

so that there is an even number of fleet vehicles per sector/quadrant of the city. We collected results for 12, 20, 32, 40, 52, and 60 fleet vehicles.

- *Number of Traffic Vehicles*: We have chosen to use a fixed number of one thousand traffic vehicles as our fluctuating average during working hours. We have chosen this number based on observations within SUMO's graphical user interface in terms of traffic congestion. With this number, we thread the line between emulating realistic traffic volume and the limitations of the converted road map that SUMO is capable of using. We did not include rush hour bursts of traffic in this model, but the occasional unintended congestion resulting in vehicle teleportation provides example periods of extra congestion that do not exceed the realm of possibilities given real-world constraints, i.e., traffic accidents, construction, etc.
- *Service Stop Duration*: To keep in line with our VDTN model, all fleet vehicles make customer service stops during working hours. We have chosen a fixed time of twenty minutes per stop. The vehicle is parked by the roadside (i.e., removed from and placed adjacent to the road so as not to impede traffic) during this time period.
- *Rest Stop Duration*: To create a worst-case scenario for a rest stop, the duration at the stop is set to two minutes, which is minimally necessary to transact quickly at the rest stop, i.e. buying a coffee.
- *Random Number Generator (RNG) Seeds*: We use three seeds to control randomness within our simulation. The purpose of using these seeds is to allow reproducibility of our results as well as maintain consistency throughout our various simulation runs. To be clear, all simulation runs use the exact same seeds. Each of the three seed is responsible for:
  - *Traffic Routes*: The order with which routes are chosen and given to traffic vehicles.
  - *Departure Times*: Each fleet vehicle is assigned a random start time that falls within a provided interval. This start time is

assigned at the beginning of each day.

* *Stop Locations*: These are the customer service stop locations. Each fleet vehicle chooses a random location on the map as their next stop location, based on this seed.

5.2.1 Example City: Lethbridge

There are two things to consider when picking an example city to simulate with SUMO. The first is simulation runtime complexity, and the other is traffic mobility efficiency. Runtime complexity can be controlled by reducing the size of the network map and the number of concurrent vehicles being simulated. In this context, the word *size* refers directly to the number of components (streets, intersections, traffic lights) contained within the network map, and not the actual physical space being represented. On the other hand, providing efficient traffic mobility can be achieved through manual alterations of network maps. These alterations can be time consuming, as they require initially to locate areas of high vehicle congestion, followed by assessment and manual corrective measures (see Section 4.1.1).

Taking these factors into consideration, we have chosen to use a small area in southern Alberta, Canada. The area includes one major city, Lethbridge, and two small neighboring towns, Coalhurst and Coaldale. According to Statistics Canada, their populations as of 2016 were 92,729, 2,623, and 8,153 respectively [49] [48] [47]. It should be noted that there may be a small additional population in the rural area connecting these three municipalities.

An overview of our example network map can be seen in Figure 5.1. In it, the locations of the depot (orange square), WSN gateways (yellow rectangles), and rest stop areas (blue squares with red borders) are shown. The rest areas provide opportunistic access to the internet through WiFi APs.

5.2.2 Rest Stop and WSN Gateway Locations

In our example city, we have chosen to use known locations of a ubiquitous Canadian fast-food chain (Tim Hortons) as our opportunistic rest stop WiFi locations. There are thirteen locations of the chain in the map of Lethbridge. To further add realism, we only allow fleet vehicles to stop at these locations

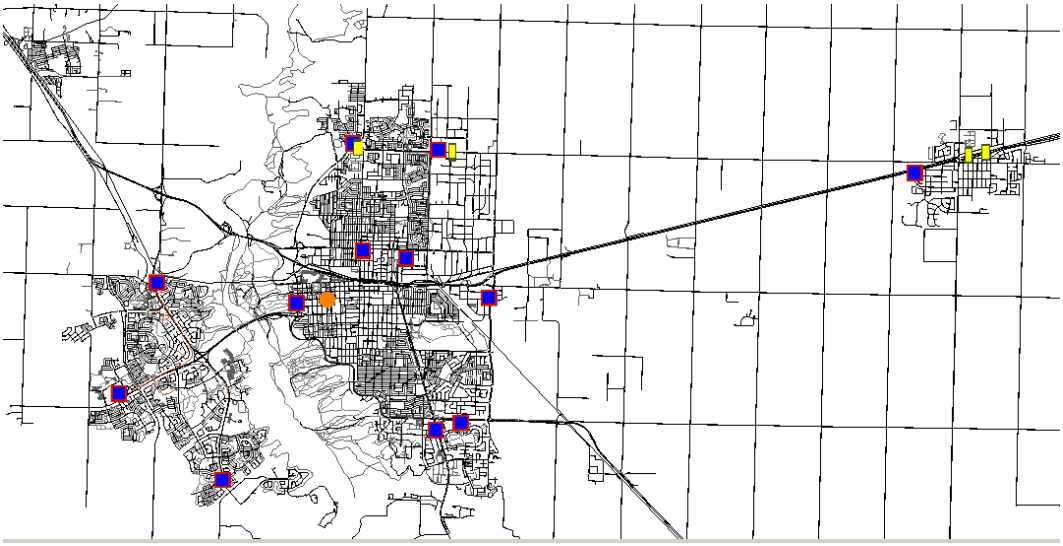


Figure 5.1: Map showing Lethbridge, Coalhurst (west), and Coaldale (east), as well as rest stops (blue squares), WSN gateways (yellow triangles), and the fleet depot (orange diamond).

under the following circumstances: 1) the vehicle must be traveling on a road adjacent to one of these rest stops, 2) the current time of day must be greater than two hours past the start of the work day, and 3) vehicles are only allowed to make one stop per working day.

The placement of WSN gateways was arbitrary and driven primarily on features of the map, i.e., close to plausible utilities corridors, depicting the possibility that the WSNs are monitoring parts of the utilities infrastructure.

5.2.3 Teleporting Exceptions

In Section 4.1.1 we introduced the SUMO-specific concept of vehicle teleportation. The length of time with which a vehicle must be considered "stuck" before being teleported further down its path is deterministic and can be defined by the user. We have chosen to use a value of 2.5 minutes, down from the default value of 5 minutes.

Teleported vehicles could be fleet vehicles or traffic. We do nothing to preempt or monitor traffic vehicles that SUMO teleports. Fleet vehicles, on the other hand, require special treatment to correct for the case when they are forced to teleport beyond a rest stop en route to their next stop. In such cases, the fleet vehicle is not rerouted back to the rest stop, rather it continues on

its way to a new stop. We do this to avoid cyclical patterns of fleet vehicles becoming "stuck" in a congested intersection while trying to repeatedly travel to a rest stop. It is important to note that instances where SUMO is forced to teleport fleet vehicles were found to be rare.

5.3 Simulation Results Analysis

Our first objective is to use the simulations to derive the distance dynamics DTMC. The DTMC is intended to capture the steady state behavior of the VDTN. Yet, some of the simulated time expresses departures from the depot at the beginning of the day and returns to the depot at the end of working hours. Those respond to a warmup and cooldown phase surrounding the behavior of the vehicles during the typical working day. Moreover, we need to, at least inspect whether the trajectories generated confirm the expected dynamics, especially insofar the assignment of each vehicle to each quadrant is concerned. Subsequently, we will examine the encounters at various distances and their dynamics that will be summarized in the DTMC.

5.3.1 Trajectory Plausibility

First, let us examine if the simulations produce valid paths. We present in Figures 5.2 and 5.3 the thumbnail sketch of the trajectories followed during a day by each of the 60 fleet vehicles, in a 60-vehicle simulation scenario. The figures should be interpreted against the map in Figure 5.1.

There are two characteristics that we notice as properly captured in the example trajectories. First, most of the trajectories are attracted within the city of Lethbridge, and fewer in the surrounding communities, as the service points visited are more densely present within Lethbridge. In a few cases, the vehicles visit the further removed communities of Coaldale and Coalhurst. Secondly, notice that each vehicle is confined to destinations in its own quadrant of the map, given the high-level scheduling described earlier which assigns, for the working day, each vehicle to service points in a specific quadrant.

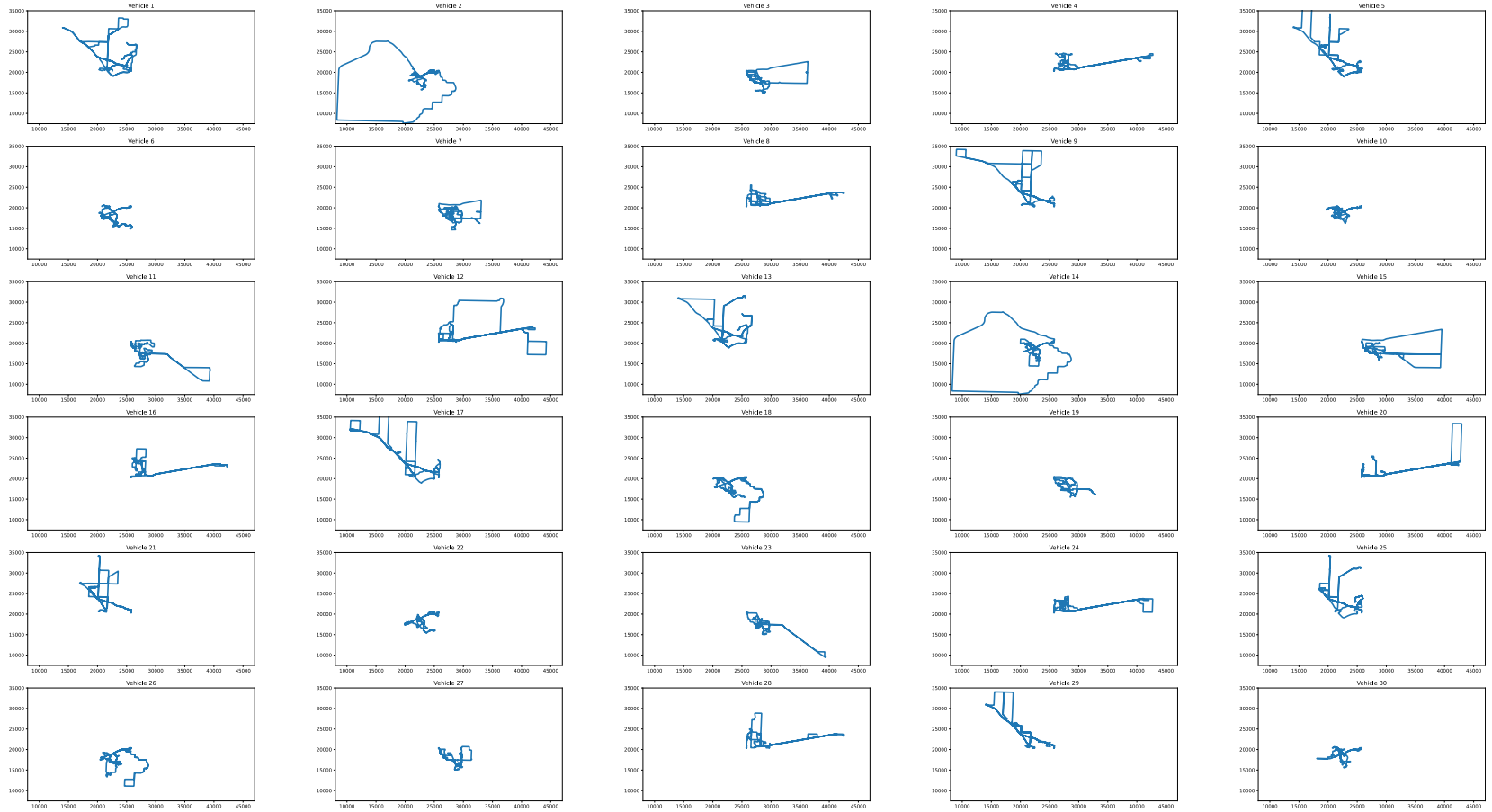


Figure 5.2: Example trajectories (vehicles 1 to 30) over an entire day.

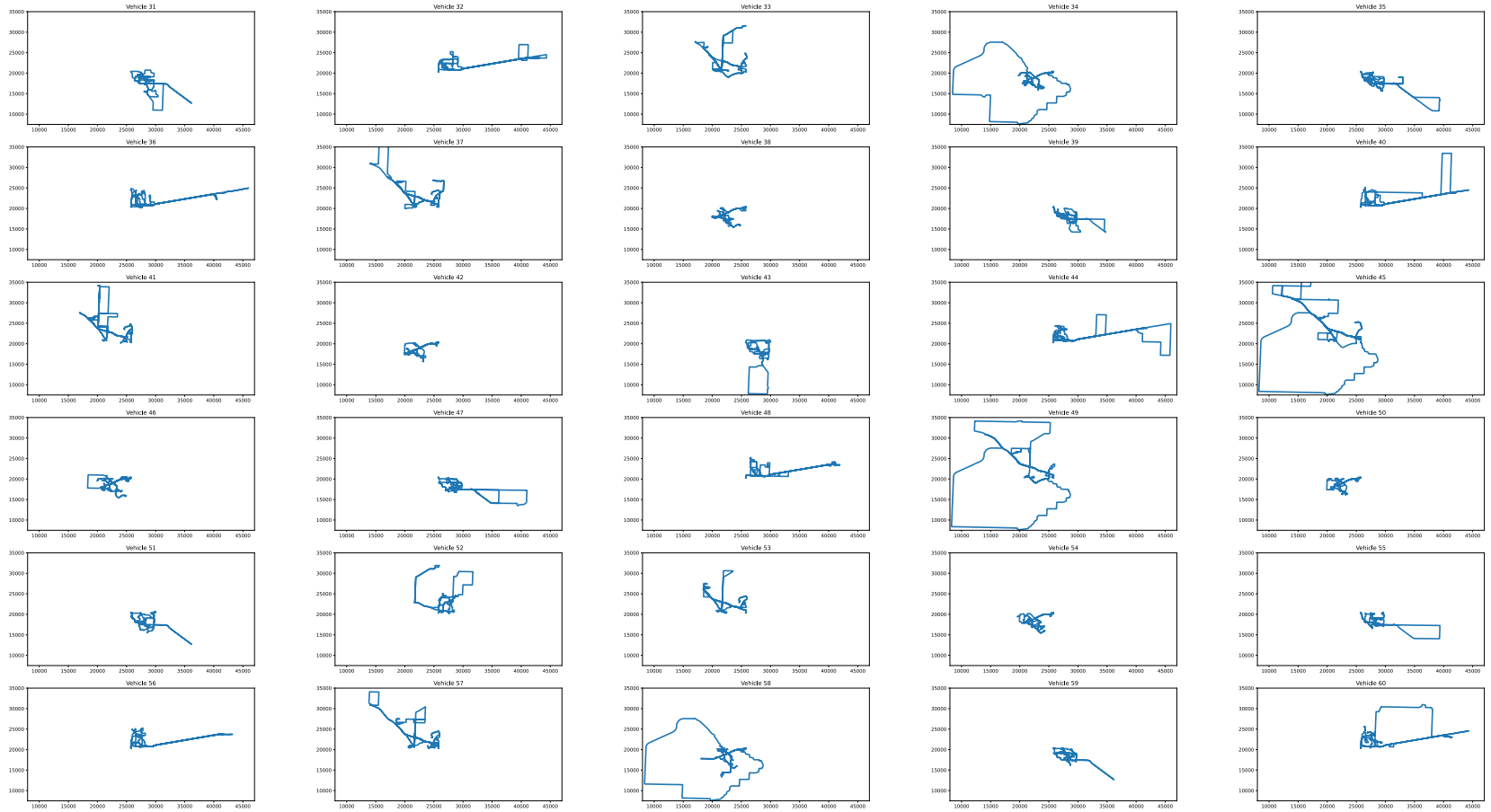


Figure 5.3: Example trajectories (vehicles 31 to 60) over an entire day.

5.3.2 Warmup and Cooldown Transients

Using our described mobility model it is possible to divide a sample workday into three subsections: warmup, steady state, cooldown. The warmup and cooldown periods describe time slices when fleet vehicles are departing from the depot in the morning and returning to the depot at the end of the workday.

Figure 5.4 indicates, for a 60 vehicle fleet, how the fraction of encounters of pairs of vehicles at a particular distance, evolves throughout the day. The top figure are encounters that manage to get closer than 10 meters, the next those that get closer than 20 meters, etc. A fraction of encounters that are achieved at a certain distance, also become, subsequently, closer distance encounters. The really close encounters (less than 10 meters) are relatively few compared to the ones that are less than 200 meters.

A notable feature is that the encounters occur more often (per unit of time) in the beginning of the shift. This is because the vehicles leave from the same depot. Even though we uniformly randomize the departure from the depot to an interval of [8am,8:10am], two (or more) departing vehicles can still be close to each other before their paths diverge. Therefore, we, rather arbitrarily, disregard the first 20 minutes of the work day (starting data collection at 8:20am) to avoid the spike on those initial encounters that are uncharacteristic of the steady state. Symmetrically to this (though not as evident in Figure 5.4) we discard the last 20 minutes of the working day, stopping the encounter statistics gathering at 4:40pm. Note that a fleet vehicle will only make another stop if there is enough time (>20 minutes) to do so before the end of the work day.

5.3.3 Encounter Dynamics

We can aggregate the encounters crossing various distance thresholds, such as the ones depicted in Figure 5.4 and capture the probability that, given that two vehicles have crossed the D_{max} threshold, they will get closer (cross a smaller threshold, at least once) during that encounter. One can think of this as the conditional probability that, having started an encounter at 200 meters, they will get to a closer interval and, presumably, will be able to communicate at higher bit rates.

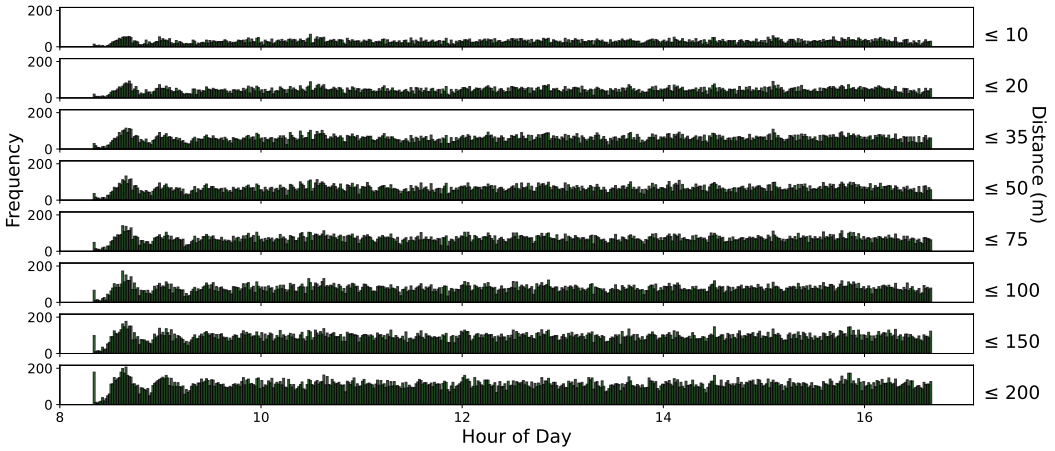


Figure 5.4: Encounter tallies at a distance less than or equal to a distance threshold vs. time of day for one simulated working day (60 vehicle fleet).

The results are encouraging and are depicted in Figure 5.5 for 12 vehicle fleet and in Figure 5.6 for 60 vehicle fleet. It appears that there is a good probability (more than 0.2) that once two vehicles start an encounter, they will also get in the closest interval of [0,10] meters, and similarly higher probabilities for further distance intervals. We also plot the confidence intervals to demonstrate that a difference between the small fleet and large fleet configurations is a slightly noticeable variability for the small fleet case. This should not be surprising as the encounter distance dynamics are driven by the topology (the map) rather than the size of the fleet. For example, when two vehicles head to the same intersection with a traffic light from difference directions, they almost certainly meet at a shorter distance than when the encounter started.

What the conditional probability does not reveal is whether, during an encounter, the sojourn time within a certain distance interval is long or not. For example, we may be able to get within 10 meters of another vehicle, but for how long? To this end we examine the duration histogram of encounters that are at least closer then a particular threshold. Figure 5.7, which is similar to figures for smaller fleet sizes, confirms a suspicion that close encounters may be common but short-lived – in the order of a few seconds for distances less than 10 meters. The further the distance considered the more likely that the nodes remain in contact for long periods of time, in the tens of seconds.

Zooming into the same kind of data, i.e., duration histogram, and looking

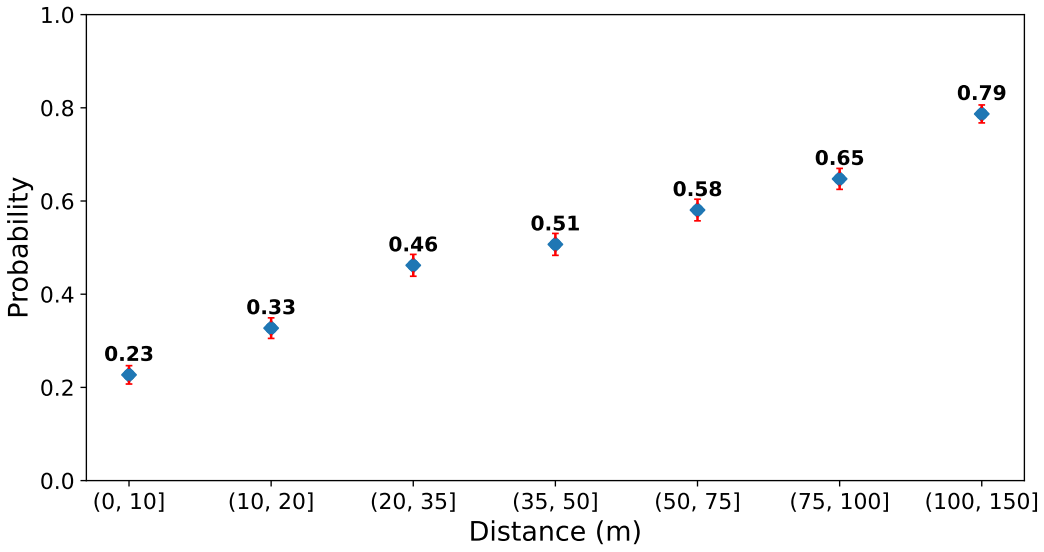


Figure 5.5: Conditional probability that, after two vehicles of a 12 vehicle fleet start an encounter, they will get, at least once, within a distance in the corresponding interval.

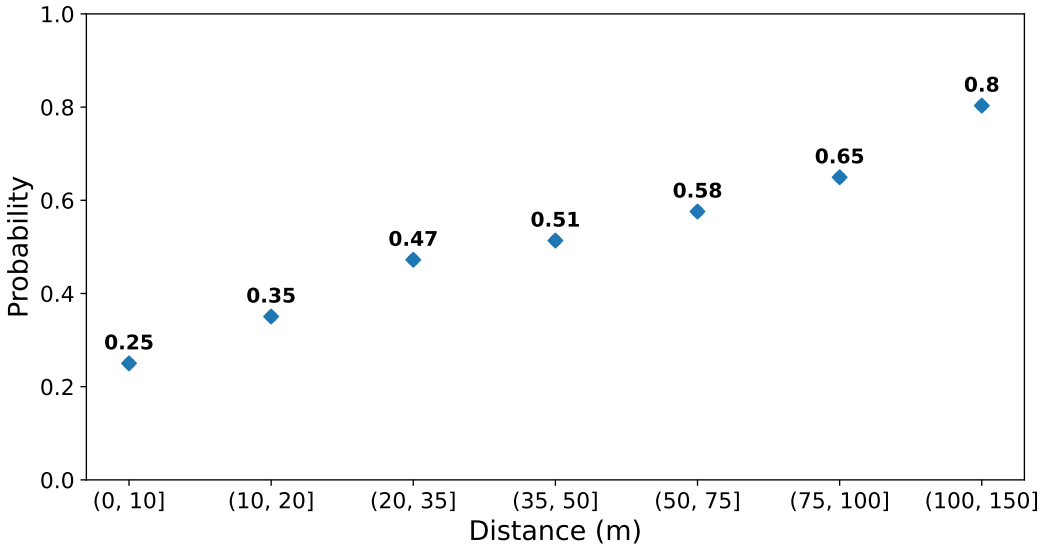


Figure 5.6: Conditional probability that, after two vehicles of a 60 vehicle fleet start an encounter, they will get, at least once, within a distance in the corresponding interval.

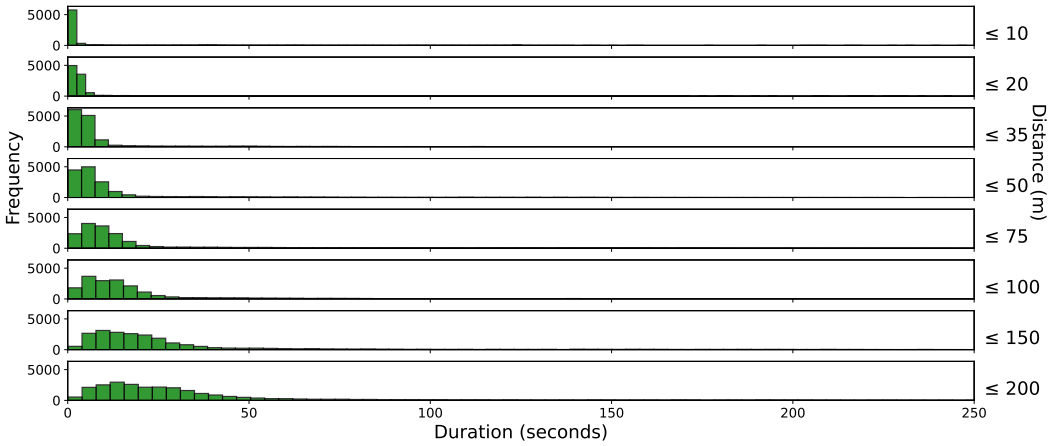


Figure 5.7: Average encounter duration for 60 vehicle fleet for encounters within a specific distance threshold.

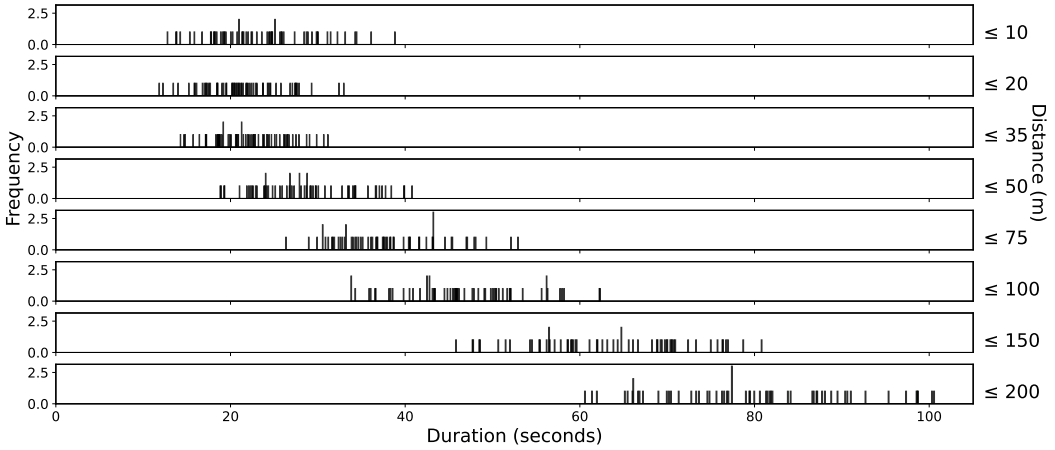


Figure 5.8: Zoomed portion of the average encounter duration for 60 vehicle fleet for encounters within a specific distance threshold.

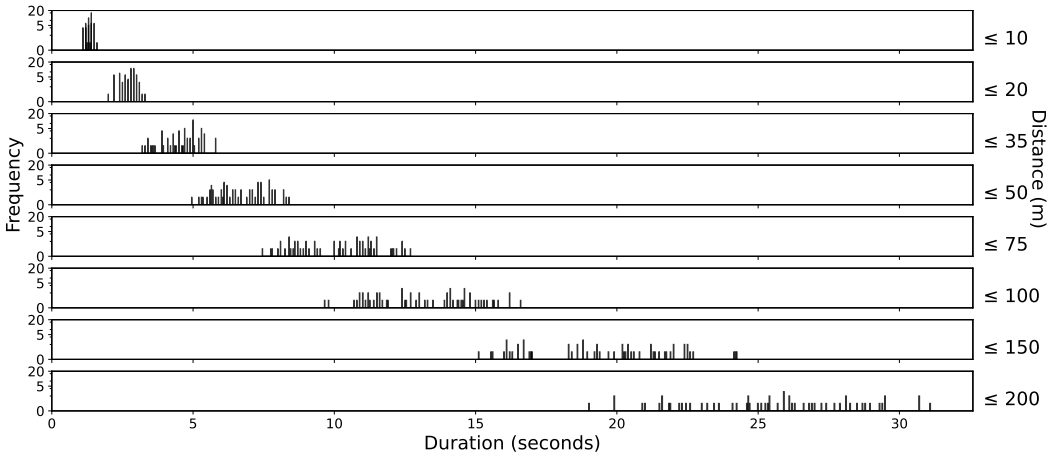


Figure 5.9: Zoomed portion of the median encounter duration for 60 vehicle fleet for encounters within a specific distance threshold.

at the individual vehicles separately (each represented as a fixed height tick, stacking the ticks when they fall on the same x value), Figure 5.8 reveals that across the population of vehicles some may be more fortunate for really long encounters even at short distances. There is the example of at least one vehicle that experienced encounters of 40 seconds at less than 10 meters. These are typically vehicles whose trajectories confine them to a highly visited area of the city, and the chances of getting close to other vehicles in that area cause them to often get close to each other, e.g., at intersections and traffic lights. Clearly, vehicles that do not orbit highly visited areas are the ones not as fortunate of long duration short distance encounters.

A final insight can be gleaned from the median, rather than the average, encounter duration below a particular distance threshold, as captured in Figure 5.9 for the same runs as those of Figure 5.8. Notice that the median encounter durations are quite concentrated at the smaller distances. Also note, considering that the x-axis of Figure 5.9 spans smaller values compared to the averages in Figure 5.8, that the medians appear to be overall smaller than the averages. This further suggests that a few "lucky" nodes can attain long duration encounters, even at small distances, while the bulk is not as fortunate.

The discrepancy between average and median suggests that, while we pursue a single Markovian model capturing a "typical" node, the variability

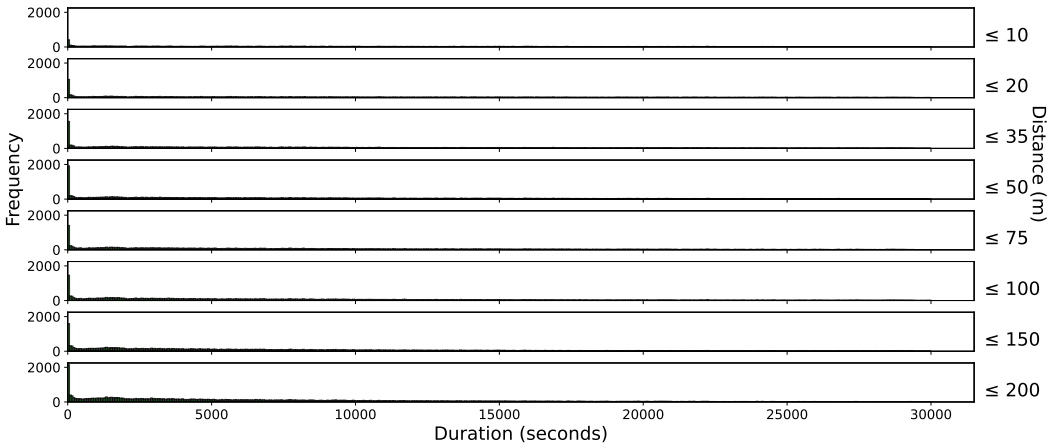


Figure 5.10: Histogram of intervals between encounters for a 60 vehicle fleet.

across the population can be considerable, and any highly simplifying model, like a single representative DTMC, should be treated with due caution. Given the fact that during a working day, a vehicle is given a particular sector of the city to serve there is no average behavior specific to the entire map, but average behaviors in each quadrant/sector. A possible extension of this work to include multiple Markovian models – one per sector – is discussed in the conclusions.

To complete the picture, we examine how much time passes between encounters for a node to get closer than a threshold distance to another node, shown in Figure 5.10. The pattern is the same across threshold distances: heavy tailed distributions indicating very long intervals between encounters crossing a specific distance, no matter the chosen distance. At long distances, there is a possibility that two nodes will again find themselves at this distance, essentially being distant enough on the map but within the same general area, so the 200 meter threshold is crossed often. Interestingly, when two nodes are very close to each other, it might still not take a lot of time until they encounter each other at the same short distance (see small peak at ≤10 meter, and more so at ≤20 meters at durations close to zero). This rather unexpected result may be the result of vehicles “trapped” close to each other, on the same road segment, possibly even travelling in the same direction that move closer then further apart, and then closer, as the traffic congestion dictates.

We conclude with an example of the introduced DTMC, whose transition

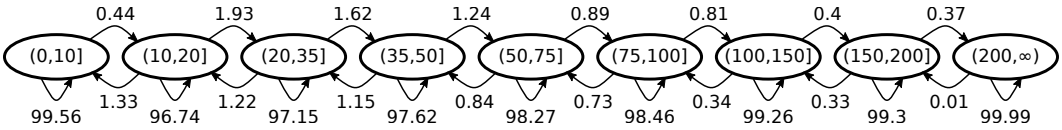


Figure 5.11: DTMC transition probabilities (as %) for a 60 vehicle fleet.

probabilities and state definitions are shown in Figure 5.11, for a 60 vehicle fleet, and for a time step of 0.1 seconds (the SUMO simulation step size). The probabilities out of each state are shown as percentages and their sum is 100% (+/- 0.01% due to rounding). The transition probabilities are similar for smaller fleet sizes. The steady state probabilities π can then be determined, and in Chapter 7 will be used to derive the VDTN data carrying capacity, C_{VDTN}. But first we need to determine, experimentally the throughput at various distances, which requires field experiments.

Chapter 6

Field Experiments

This chapter presents how field experiments were carried out to measure throughput between two endpoints in an outdoor setting and at various distances. The platform used allowed us to perform IBSS measurements using a 40 MHz channel in the 2.4 GHz band and Infrastructure measurements using a 80 MHz channel in the 5 GHz band. The communication was carried out with wireless interfaces operating in compliance to the IEEE 802.11ac standard. The obtained performance measurements are subsequently used to provide plausible throughput for VDTN communication. In the following we describe, the hardware and software platform used for the measurements, and how we handled some of the platform limitations, followed by a description of the measurement methodology.

6.1 Hardware and Software Platform

The communicating peers are Dell Wyse Dx0Q Thin Client computers, with quad-core AMD GX-415GA 1.5GHz CPU, 8 GB RAM, and 16 GB Flash Storage. The thin clients ran Linux, from the Alpine distribution (version 3.13, Linux kernel 5.10.16-0-lts). Each of the thin clients had two WiFi devices: an internal Intel AX200 (mini PCIe, firmware microcode version `59.601f3a66.0 cc-a0-59.ucode`, `iwlwifi` Linux driver) WiFi 6 (IEEE 802.11ax capable) card with two physical external antennas, and an external USB WiFi dongle, based on the Ralink RT5572 (IEEE 802.11n capable). The AX200 interface was used for all the measurements, while the RT5572 was used for communicating with the two peers for user interaction during the experiments, e.g., for starting a

new measurement script etc. To avoid any interference in the measurements, the RT5572 dongle was operating at a separate channel (channel 11, 20 MHz wide) from the channels used for the throughput measurements. A laptop was utilized for connecting to both thin clients via the RT5572 dongles.

For infrastructure mode experiments, one of the thin clients configured its AX200 as an AP using `wpa_supplicant` (version 2.8) and `hostapd` (version 2.9). With IBSS we made use of a lightweight open source networking tool called `iw` (version 5.9) which is capable of changing the mode of the internal network card to IBSS, as well as establishing and maintaining a connection with another peer device.

`iperf3` (varsion 3.9), a free throughput measurement tool was used as it is capable of testing both TCP and UDP connection types. It reported the throughput once per second. At the same time that `iperf3` was performing throughput measurements, `iw` was periodically invoked to record the various connection parameters while throughput testing was being conducted. Of the information recorded by `iw` we were particularly interested in the reported bit rates and Modulation Coding Scheme (MCS) values, which help interpret the throughput measurements via the limitation placed on bit rate depending on the distance between the two peers and the behavior of the device's rate adaptation algorithm.

6.1.1 Platform Limitations

The intention was to use the maximum possible channel width (160MHz) in the 5 GHz band since the AX200 device is supporting the IEEE 802.11ax standard (known as "WiFi 6") which, with two antennas, can attain a theoretically possible maximum bit rate of 2.4 Gbps on a 160MHz wide channel (which is only possible in the 5 GHz band). Naturally, some fraction of the available bit rate will be sacrificed to overheads, e.g., to protocol header overheads, but the intention was to achieve close to those high rates at very near distances. In all experiments we were never able to exceed the 80 MHz channel width in the 5 GHz band. Effectively, the "ax" operation (enabled via `ieee80211ax=1` option in `hostapd.conf`) was unsuccessful, and, instead, a maximum of 80 MHz channel width was the maximum achieved.

Band	Ch.	Freq.	Usage	maxEIRP	Restrictions
UNII1	36	5.180	Indoor	200	None
	40	5.200			
	44	5.220			
	45	5.240			
UNII2	52	5.260	Indoor & Outdoor	1000	DFS
	56	5.280			
	60	5.300			
	64	5.320			
UNII2 Extended	100	5.500			
	104	5.520			
	108	5.540			
	112	5.560			
	116	5.580			
	120	5.600	Restricted	N/A	Canadian Weather Radar
	124	5.620			
	128	5.640			
	132	5.660	Indoor & Outdoor	1000	DFS
	136	5.680			
	140	5.700			
	144	5.720			
UNII3	149	5.745	Indoor & Outdoor	4000	None
	153	5.765			
	157	5.785			
	161	5.805			
	165	5.825			

Table 6.1: Channels available for WiFi use in the 5 GHz band in Canada.

The source of this limitation appears to be the interaction of two factors: the way in which AX200 observes country-specific regulatory restrictions, and the particular regulatory restrictions in Canada (Section 6 of the [25]). Table 6.1 lists all available channels for WiFi use in the 5GHz band in Canada. Frequencies are shown in GHz and are the center frequency of 20MHz-wide channels. DFS stands for Dynamic Frequency Selection and maxEIRP is the maximum allowed Effective Isotropic Radiated Power of any transmitter plus antenna system, and it is listed in milliWatts.

For the purposes of the WiFi standard specification, allocating a 160 MHz channel is equivalent to allocating eight consecutive 20 MHz channels or two separate sets of four 20 MHz channels (called "80+80"). Note from Table 6.1, that no eight consecutive, or two groups of four each, channels exist without either, (a), being different in at least one of the use characteristics (usage, maximum EIRP, and restrictions), or, (b) requiring Dynamic Frequency Selection (DFS). DFS requires that the device senses the channels for adequately long period of time, and keeps monitoring them even when found free, to ensure no other transmitter is detected there. The DFS strategy is used to avoid creating interference to non-WiFi services such as weather radar. DFS requires that a correspondingly elaborate channel assessment logic is implemented by the device. In the absence of DFS implementation, the device is not expected to operate in the channels requiring DFS. A device unable to implement DFS, in Canada, is left with two "islands" in Unlicensed National Information Infrastructure radio bands UNII1 and UNII3 that are different in terms of use scenarios (indoor/outdoor) and emitted power. In the absence of any information as to whether a device is going to be used indoors or outdoors, the most conservative strategy that satisfies the regulatory constraints is to favor a single 80 MHz channel in UNII3. Consequently, in Canada, a 160 MHz outdoor channel is not a realistic expectation for devices lacking DFS capability.

The second part of the puzzle is a disclosure [21] made by Intel to the Federal Communications Commission on the way it implemented regulatory compliance in its AX200 products. An important quote from the letter is the following:

"The device operates as a client without radar detection capability and will be programmed at the factory to passively scan on the following dynamic frequency selection (DFS) channels and will only listen for a master device and cannot send a probe request to initiate communication on these DFS channels. Accordingly passive scanning provides protection for TDWR operations and preventing transmission in the 5600MHz – 5650MHz frequency band. Client software and drivers will never enable the device to act as a master or GO for operation in DFS frequency bands and therefore ad hoc mode is always disabled on these passive scan DFS channels."

The letter goes on to list the specific channels and bandwidth options where the above apply. Note that the summary of the above points is that, in DFS channels, the AX200 will never initiate a transmission on its own, but only in response of another WiFi node already transmitting there (e.g., another already operating AP). Initiating a transmission is essential for e.g., acting as AP, since APs should be able to send beacons. Similarly, ad hoc (IBSS) cannot happen either. Next the letter explains that:

"This device meets the requirements of FCC Part 15.202 and accordingly will be programmed at the factory to active scan on the following non-DFS channels to initiate communication during normal WLAN operation. When operating in WiFi Direct mode on these non-DFS channels, it may operate as a P2P client device or GO to establish a P2P network if, and only if, a master device is present and network communication is maintained between a master device and the GO."

This excerpt suggests that in non-DFS channels, the AX200 can transmit first if it operates as a client (hence the "active scan") in search of e.g., an AP. However, this restricts it to assuming the role of a client device. It still does not endow the device with the possibility to act as an AP.

A conjecture as to why AX200 behaves the way it does is that, first, the manufacturer does not have to deal with the complexity of implementing DFS. DFS compliance is therefore delegated to another device in the same area

(not an AX200) being present and implementing the DFS logic and, e.g., acts as an AP (then the AX200 can connect to this AP). A way to conceptualize this design decision is to think that the AX200 is passively listening to the channels to see if other WiFi devices have taken steps to assume use of the channels. If yes, then those channels should be OK for the AX200 to use them as well. Mechanisms of this sort are sometimes called *self-managed* regulatory domain management. Self-managed devices, upon power-on have no information about which country and regulatory constraints they operate on, and hence assume the most restricted scenario insofar transmissions in the 5 GHz band are concerned. Subsequently the AX200 overhears APs in some of the channels and "opens" those channels for operation. We speculate that the AX200 may also be using location information (country information) present in the, overheard, beacons of nearby APs to determine the country where it is operating and the corresponding regulatory restrictions. However, we did not conduct experiments geared to confirming or refuting this point.

Circumventing the Limitations

The inability of AX200 to operate as AP in the 5 GHz band has been discussed in the broader user/developer community and the best solutions so far (such as the one reported in Stack Exchange [18]) appear to be exploiting what is probably a form of race condition via the device driver. While AX200 is probing (scanning) as client for an AP with a non-existing SSID in the 5 GHz channels using, for example `wpa_supplicant`, it "opens" access to the non-DFS channels. The process of scanning is continuously unsuccessful in locating the particular SSID. If, simultaneously, on the same AX200 we instantiate an AP on a non-DFS channel, using `hostapd`, it appears that the AP will start successfully, and once this has happened, the scanning of `wpa_supplicant` can be stopped without impacting the AP's operation. The simultaneous use of the device as both a client that is scanning and as an AP requires the existence of two device interfaces. Fortunately, the device driver supports "virtual interfaces" (with their own individual device identifiers) so one additional interface can be created by referring to one and the same device, in addition to the interface created by default.

We followed the workaround described in [18]. Additionally, after extensive testing, it was not possible to allocate a 160 MHz channel. We resorted to allocating 80 MHz channels through the workaround. Moreover, the "ax" option was unsuccessful, hence we set the `hostapd` to supporting up to IEEE 802.11ac (`ieee80211ac=1`). Our workaround and compromise in terms of bandwidth and IEEE 802.11 standard version, limits the higher bit rates achievable to those listed in Table 6.2. The rates shown in the table are in addition to the legacy IEEE 802.11g data rates that are also supported. The bit rate depends on the channel bandwidth (20, 40, and 80 MHz), the guard interval (GI) length (0.8µs and 0.4µs), the number of spatial streams (SS), and the modulation scheme used (not shown in the table). The combined effect of SS and modulation is summarized by the modulation and coding scheme (MCS) index. HT (High Throughput) indices were defined by the IEEE 802.11n standard and VHT (Very High Throughput) indices where defined by the IEEE 802.11ac standard.

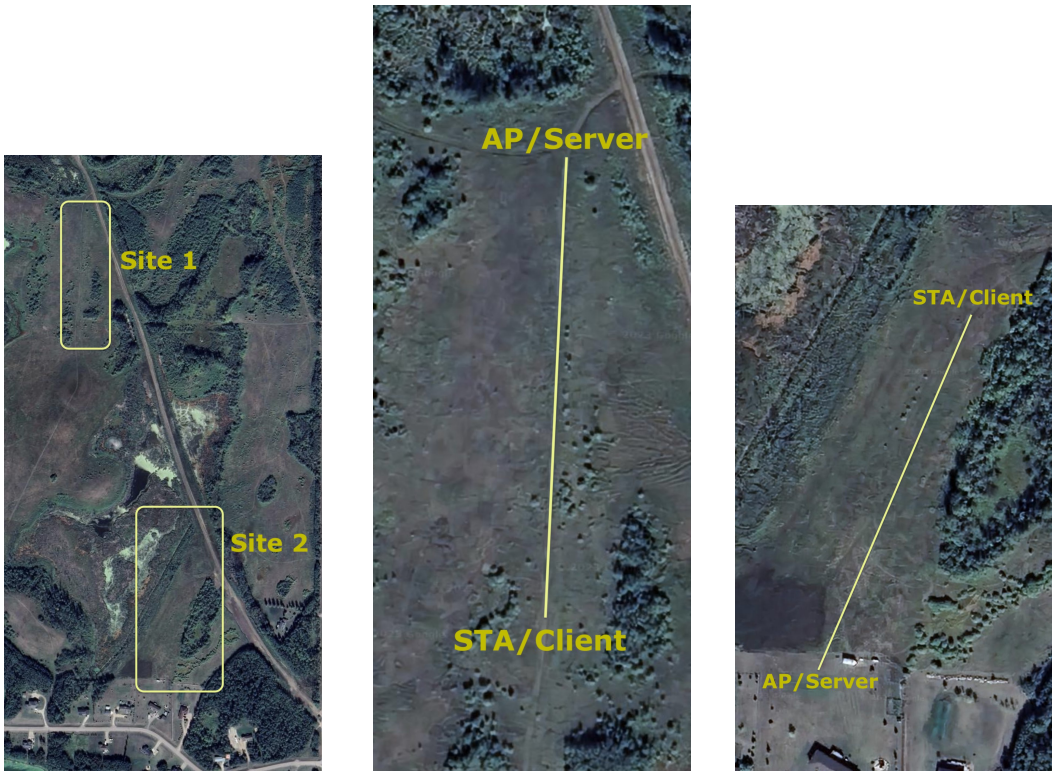
Notice that the workaround presented in [18] that we followed is not immediately translateable to the case of IBSS communication and we did not invest effort into producing a workaround for IBSS. The operation of IBSS mode in the 5 GHz mode is a highly debatable issue anyway since, due to regulatory constraints, one can never be sure whether a device can initiate transmissions in a 5 GHz channel, and the vendors are showing very little interest to supporting IBSS at 5 GHz. As far as our experiments are concerned, all our IBSS experiments were carried out in the 2.4 GHz band and with a 40 MHz bandwidth which is the maximum possible in that band.

6.2 Location

All wireless testing was conducted at two sites near Cold Lake, Alberta on private property. An aerial shot of both sites is shown in Figure 6.1. At the very bottom of 6.1(a) there are dwellings, in an otherwise uninhabited land. Site 1 is much more isolated than site 2, as can be seen in figures 6.1(b) and 6.1(c). Due to its relative seclusion, two portable generators were used to power both thin clients. For site 2 only one generator was needed to power the mobile thin client. Due to relative close proximity to a house, an extension

MCS Index			20MHz		40MHz		80MHz	
HT	VHT	SS	0.8µs	0.4µs	0.8µs	0.4µs	0.8µs	0.4µs
0	0	1	6.5	7.2	13.5	15	29.3	32.5
1	1	1	13	14.4	27	30	58.5	65
2	2	1	19.5	21.7	40.5	45	87.8	97.5
3	3	1	26	28.9	54	60	117	130
4	4	1	39	43.3	81	90	175.5	195
5	5	1	52	57.8	108	120	234	260
6	6	1	58.5	65	121.5	135	263.3	292.5
7	7	1	65	72.2	135	150	292.5	325
	8	1	78	86.7	162	180	351	390
	9	1	N/A	N/A	180	200	390	433.3
8	0	2	13	14.4	27	30	58.5	65
9	1	2	26	28.9	54	60	117	130
10	2	2	39	43.3	81	90	175.5	195
11	3	2	52	57.8	108	120	234	260
12	4	2	78	86.7	162	180	351	390
13	5	2	104	115.6	216	240	468	520
14	6	2	117	130	243	270	526.5	585
15	7	2	130	144.4	270	300	585	650
	8	2	156	173.3	324	360	702	780
	9	2	N/A	N/A	360	400	780	866.7

Table 6.2: Attainable bit rates, in Mbps, by the used equipment and setup.



(a) Both sites (b) Site 1 (c) Site 2

Figure 6.1: Aerial view of the field test sites.

cord was used to power the stationary thin client. Additional photographs of the area and setup are provided in Appendix A.

6.2.1 Site 1 vs. Site 2

Due to its relative remoteness, Site 1 had no nearby interfering WiFi networks that could be found through scanning. At site 2, however, there did exist a few networks that could be detected from the position of the stationary thin client at the edge of the house yard. There was only one nearby WiFi network operating in the 5 GHz band on channel 44 (80 MHz wide). We chose channel 149 when performing Infrastructure measurements to ensure no interference with the single pre-existing network. The WiFi networks in the 2.4 GHz band were operating on channels 4, 6, 7, 10, and 11. We chose channel 1 with 40 MHz bandwidth for the IBSS mode experiments to minimize overlap with the pre-existing networks on channels 4, 6, and 7 and no overlap with those on channel 11. The pre-existing networks were intermittently active and, at the

time of our field measurements, there were good indications those networks were not being used to any significant degree.

Site 1 was chosen initially due to its relative seclusion from any sources of interference, and hence similar to what one might encounter in a segment of isolated highway far from buildings. This was the initial site for our testing, and indeed all of the IBSS results, save for a few retests, were conducted there. There was a problem with obtaining and maintaining an AP using infrastructure mode, which forced us to find a new location (Site 2) that was just close enough to other APs. This experience seems to support the speculation that the AX200 needs the presence of other APs (and possible country code information in beacons) to be coaxed into operating as an AP.

6.3 Measurement Methodology

Our field measurement procedure was informed by the simulation strategy described in previous chapters. As such, we chose as measurement distances the distances that correspond to the midpoints of the intervals used in the simulation and Markov chain analysis in previous chapters; that is: 5m, 15m, 27.5m, 42.5m, 62.5m, 87.5m, 125m, and 175m. We additionally measured performance at 200m as this represents the threshold distance beyond which, we consider vehicles to be "not close enough to transmit reliably" in our simulation model. Each device was placed on a table with clear line of sight of each other. One device was stationary while the other one was moved to the various testing locations. Each of these locations was marked previously using a wheel tape measure and wooden stakes driven into the ground.

At each location `iperf3` was run for exactly one minute while collecting diagnostic information with `iw` simultaneously. This procedure was repeated to cover both TCP and UDP traffic using both Infrastructure (AP) and IBSS mode. Moreover,

- all Infrastructure mode measurements were performed using channel 149 (5 GHz band) and channel width of 80 MHz, and,
- all IBSS mode measurements were performed on channel 1 (2.4 GHz band) and channel width of 40 MHz.

The commands for performing the measurements were variations of:

```
iperf3 --timestamps -s -B 10.229.1.1 &> iperf_out_125
iperf3 --timestamps -c 10.229.1.1 -t 60 &> iperf_out_125
iperf3 --timestamps -c 10.229.1.1 -t 60 -u -b 0M &> iperf_out_125
```

The first is an example of the command running at the server, awaiting the connection from a client. The second is example of the command running on the client for 60 seconds of TCP measurements, and the last is an example of a command running on the client for 60 seconds of UDP measurements. File names of recorded measurements include the distance (125 meters in the above examples) they correspond to. The `-b 0M` option removes rate limits from UDP traffic. UDP payloads are the default 1470 bytes. TCP adjusts its rate via the dynamics of the TCP transmission window. The bit rates were recorded using the `link` option of the `iw` command.

Chapter 7

Results

We present the various measurements collected from our field experiments. As expected, the further the distance, the lower the throughput. We relate the throughput measurements with the continuous rate adaptation taking place at the transmitter which selects the best coding rate to match the observed channel conditions. We consider the following factors: Distance vs. Throughput, UDP vs. TCP, and IBSS vs. Infrastructure (AP). Armed with DTMC's steady state probabilities, π and the throughput measurements, b_i, we proceed to calculate the total VDTN data carrying capacity $C_{VDTN} = \sum_i \pi_i b_i$.

7.1 Throughput vs. Distance

The fundamental observation, on which the field experiments are motivated, is the change of attained throughput with respect to distance. The further the distance the weaker the signal received, the lesser the signal "stands out" over noise, i.e., the lower the Signal-to-Noise Ratio (SNR). The IEEE 802.11 standard is a standard that explicitly accommodates for a deteriorating (or improving) SNR by decreasing (increasing) the bit rate at which the transmitter is sending data. Higher bit rates are possible at higher SNR, lower bit rates at lower SNR. The transmitter achieves this adaptation via changes to the the modulation and coding strategy that it uses. The receiver (and the frame format) has been designed such that the receiver can quickly determine the modulation scheme used and decode, as needed, the transmitted frame. Of course, beyond a certain distance, the SNR is so poor that the lowest bit rate modulation is unable to ensure the receiver receives the transmission with

any reasonably high probability. Multiple antennas improve the ability to receive weak signals and are also used to improve bit rate using coding that exploits the antenna diversity.

An IEEE 802.11ac compliant transmitter, as the one used in the experiments, with two spatial streams (antennas) can, theoretically, attain the rates shown in Table 6.2 along with the legacy rates of the IEEE 802.11g standard (6, 9, 12, 18, 24, 36, 48 and 54 Mbps, which also include 1, 2, 5.5, and 11 Mbps from the even earlier IEEE 802.11b). Hence, when reception is possible, the lowest bit rate one can expect is 1 Mbps. What determines the bit rate chosen at any point is the rate adaptation algorithm executed at the transmitter. Remarkably, the rate adaptation strategy is **not** included in the IEEE 802.11 standard! This has led researchers to various proposals of the best rate adaptation but one of the most popular rate adaptation has been the `minstrel` [30, 56, 58] algorithm. Variants of `minstrel` were also developed, e.g., [17], as higher bit-rates became available in subsequent extensions of the IEEE 802.11 standard. However, Intel has been known to implement a different rate adaptation, which has been reverse-engineered in recent years [20], and through simulations it was found to be similar in performance to `minstrel`, at least insofar static environments (no fading, no mobility) was observed.

While we will not look into the specifics of Intel's rate adaptation strategy, the conclusion is that: (a) it is totally possible for the exact same experimental conditions to get a different bit rate using different device hardware, and (b) we expect that the dynamics of the rate adaptation algorithm can result in variations of the bit rate even if the experimental conditions are fixed. Point (b) reflects the strategy employed by rate adaptation algorithms to attempt to, briefly, transmit at a higher bit rate ("probing" for a higher rate), only realizing that it is not possible to do so successfully, and then resorting to a lower rate, only to repeat the process later on.

A clear example of the throughput vs. distance tradeoff can be seen in Figure 7.1 as measured from the client (sender) side with UDP payloads in Infrastructure (AP) mode. The visual language used in this and similar subsequent figures is as follows: the throughput reported by the client (sender) by `iperf` is shown as blue crosses and line plot. The *transmitting* bit rates

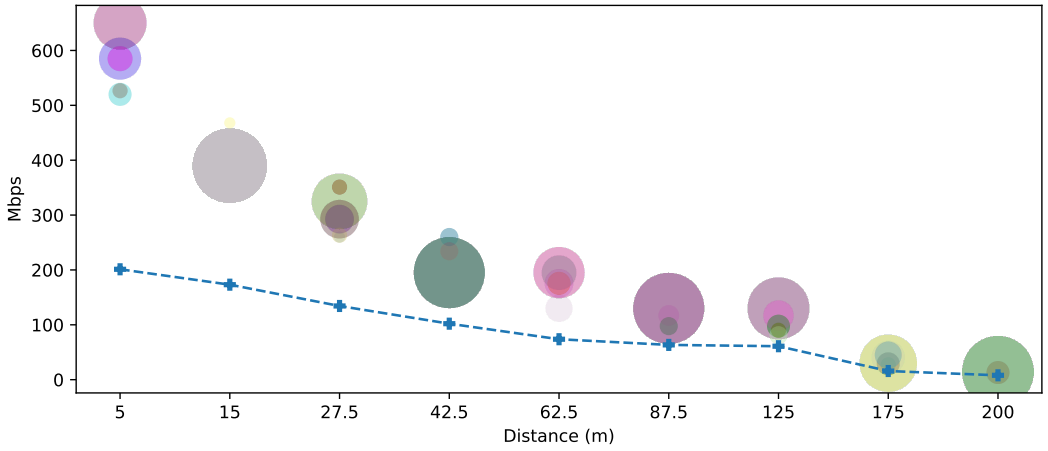


Figure 7.1: Average throughput, b_i, (blue crosses), and bit rates (circles), vs. distance; UDP payloads in AP mode.

sampled by `iw` for the corresponding distance are shown as circles. A circle for a particular bit rate is centered at the bit rate value (y axis) and its radius is proportional to the number of samples collected for that particular bit rate during the experiment. Note that the bit rate samples are *not* collected for each transmission but collected periodically and hence reflect the transmission rate that was in effect at the sampling instants. For better legibility the circles are semi-transparent and have different colors. The circles are therefore a visualization (based on periodic sampling) of the rate adaption algorithm dynamics.

For example, in Figure 7.1, at a distance of 5 meters, the rate adaptation was often (top circle) attempting to transmit at 650 Mbps (VHT-MCS 7 80MHz). Incidentally, the 650 Mbps was the largest bit rate we observed across all experiments and from Table 6.2 it is in the lower end of the rightmost column. The two higher bit rates of that column (780 and 866.7 Mbps) were not observed, but we did not measure throughput for distances closer than 5 meters where those bit rates might have been possible. The nature of the applications (V2V and V2I communication) do not justify very short distances. Moreover, due to the periodic sampling of the transmission bit rates, we are note guaranteed to have recorded all the bit rates that were attempted.

Nevertheless, it is evident that at five meters, the throughput is a small part, close to 30% (approximately 200 Mbps) of the average bit rate. At longer distances, the rate adaptation meanders around smaller bit rates, as expected, and the throughput becomes a larger percentage of the bit rate. Figure 7.2 captures the dynamics for TCP. In all cases, the bit rates form clusters, which is a sign that the rate adaptation algorithm is tracking the best bit rate over a narrow range of values.

7.1.1 A Note on Overheads

The throughput will always be smaller than the channel bit rate. In the absence of competing traffic on the channel, the throughput is impacted by overheads due to the header information attached by the various layers as well as the transmission timing constraints imposed by the medium access protocol. At higher bit rates, while the payload transmission time is reduced, not all of the overheads e.g., for medium access, are scaled by the same factor. This inflexibility of some of the medium access overhead is the cost paid by IEEE 802.11 to allow compatibility with legacy (earlier) versions of the standard. Hence, while the useful (payload) transmission time is decreased at higher bit rates, many overheads stay the same, and therefore the maximum throughput attainable is a smaller fraction of the bit rate. In short, the maximum throughput does not increase proportionately to the bit rate. Consumer-grade equipment vendors, concerned about the user expectations, warn their users of such caveats. For example, Apple [2] suggests 36% of the bit rate as realistic throughput expectation for IEEE 802.11ac and 80 MHz bandwidth (presumably at distances even shorter than the minimum of 5 meters in our experiments). Others, e.g., [55], indicate that under favorable circumstances, one should not expect a maximum throughput of more than 50% of the bit rate.

7.2 TCP vs. UDP

A comparison between UDP (Figure 7.1) versus TCP (Figure 7.2) demonstrates that, while the behavior is similar, the absolute performance differs.

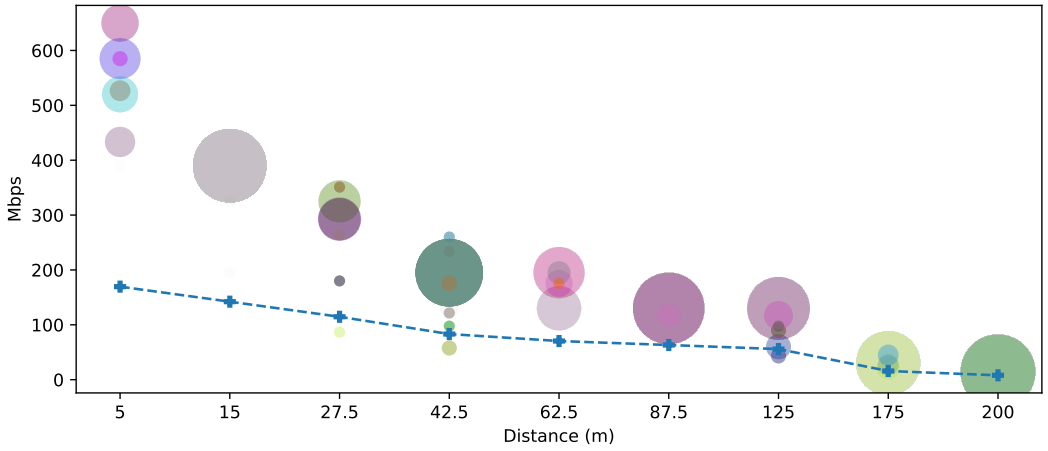


Figure 7.2: Average throughput, b_i, (blue crosses), and bit rates (circles), vs. distance; TCP payloads in AP mode.

For example at 5 meters, TCP achieves 169.2 Mbps while UDP achieves 201.2 Mbps. The difference between TCP and UDP performance is also present in all other measurements where the conditions were otherwise the same. TCP's performance deficit versus UDP is well understood to be the result of the overhead for acknowledgements, window-based congestion control, and retransmissions, especially on less reliable channels as is the case in wireless channels.

Both Figures 7.1 and 7.2 involve infrastructure mode operation whereby the AP plays the role of the server (receiver) as far as `iperf3` is concerned, and the client (sender) is the node connecting to the AP. The scenario is also relevant in V2V environments because the vehicle OBU can play the role of a VDTN peer, presenting itself as an AP to which another OBU peer can connect. Hence, in the remainder of this chapter, we do not assume that P2P OBU interaction is synonymous with IBSS mode, but P2P OBUs can also potentially communicate in a fashion where one is the AP.

In AP mode communication, the bottleneck of the data transfer is expected to be the bit rate of transmissions from the client, since it sends the data payloads used for the measurements. The transmission rate from the server to the client is less consequential as it is either not used at all (UDP) or just

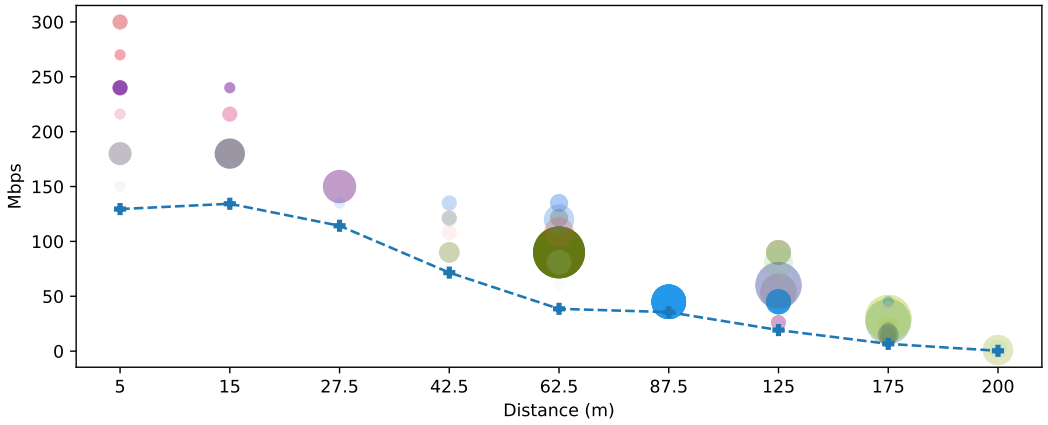


Figure 7.3: Average throughput, b_i, (blue crosses), and bit rates (circles), vs. distance; UDP payloads in IBSS mode.

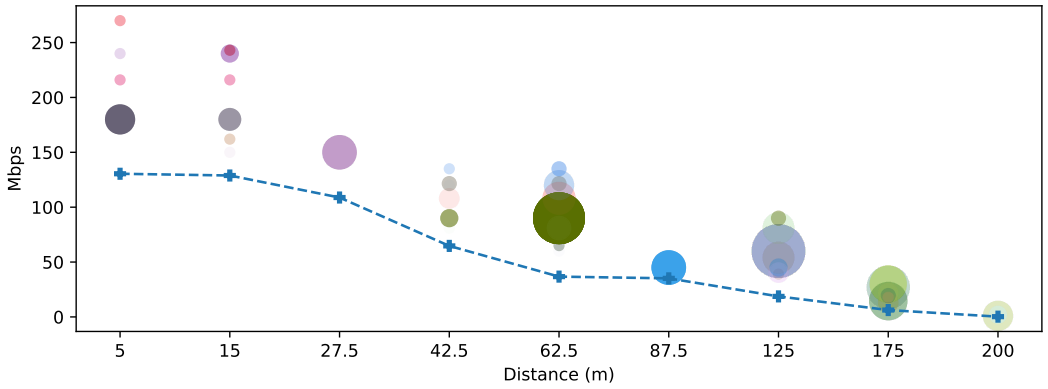


Figure 7.4: Average throughput, b_i, (blue crosses), and bit rates (circles), vs. distance; TCP payloads in IBSS mode.

for acknowledgments (TCP). Acknowledgments are short and do not require a high bit rate. Hence, the only bit rates at the client side are reported.

7.3 Infrastructure (AP) vs. IBSS

In order to measure the performance in IBSS mode, we had to abandon the 5 GHz band. As noted in earlier chapters, the flexibility of P2P IBSS mode would appear to be a good match for our application scenario, especially insofar V2V communication is concerned. However, the restriction imposed by the Initiate Radiation (IR) flag was found to prohibit a pair of nodes from starting an IBSS network and it was not possible to bypass the restriction using the same strategy (exploiting a potential race condition) that we used to bypass the same restriction for starting an AP. The most serious implication of not operating IBSS mode in the 5GHz band is the inability to use higher bit rates (higher MCS indices). In the presented results the IBSS is always carried out in the 2.4GHz band and specifically on channel 1. Thus the 5GHz band, and hence the higher bit rates are available in the AP mode.

Comparing and contrasting Figures 7.1 and 7.2 against Figures 7.3 and 7.4, the lower performance of IBSS, now limited to a 40 MHz channel and lower coding rates is evident. The bit rate adaptation algorithm again does its best to adjust the bit rate within a range. Confined to lower bit rates and in an interference free environment, the performance between TCP and UDP is difficult to distinguish but UDP still exhibits a slightly better performance.

A summary of all the average throughput (b_i) measurements is shown in Table 7.1. Note that the measurements were carried out at the midpoints of the intervals used to represent the states of the DTMC. We assume that the same b_i value applies to the entire corresponding interval. Moreover, the table shows the throughput measured exactly at D_{max} of 200 meters. A surprising realization is that while one was well advised to use 200 as virtual cutoff point in the past, modern 5 GHz IEEE 802.11ac network interfaces enable us to attain throughput in the order of 7 Mbps even at that distance. The assumption of a cutoff at approximately 200 meters is more in line with what one expects in the 2.4 GHz band and modulation. Notice that the 0.2 and 0.3 Mbps at 200 meters for IBSS might be even lower if 20 MHz channels

		Distance (m)								
Mode	**Proto**	**5**	**15**	**27.5**	**42.5**	**62.5**	**87.5**	**125**	**175**	**200**
AP	**UDP**	201.2	173.1	134.3	102.1	73.7	63.4	60.9	15.8	7.9
AP	**TCP**	169.2	141.8	114.3	82.7	70.0	62.8	55.4	15.5	7.7
IBSS	**UDP**	129.5	134.2	114.2	71.5	38.5	35.5	19.1	6.6	0.3
IBSS	**TCP**	130.0	128.4	108.4	64.3	36.3	34.7	18.2	6.1	0.2

Table 7.1: Average throughput, b_i, in Mbps as reported from the `iperf3` client end, versus distance between the two endpoints.

are used.

7.4 Determining C_{VDTN}

	Number of Fleet Vehicles					
State	**12**	**20**	**32**	**40**	**52**	**60**
(0,10]	0.00182	0.00271	0.00499	0.00494	0.00782	0.00935
(10,20]	0.00027	0.00063	0.00127	0.00177	0.00230	0.00310
(20,35]	0.00066	0.00123	0.00229	0.00337	0.00440	0.00489
(35,50]	0.00073	0.00216	0.00320	0.00373	0.00531	0.00693
(50,75]	0.00173	0.00317	0.00562	0.00639	0.01048	0.01021
(75,100]	0.00341	0.00425	0.00754	0.00890	0.01064	0.01239
(100,150]	0.00435	0.00874	0.01657	0.02013	0.02564	0.02938
(150,200]	0.00682	0.01137	0.01808	0.02497	0.03026	0.03560
(200, ∞]	0.98020	0.96574	0.94044	0.92579	0.90314	0.88816

Table 7.2: DTMC steady state probabilities, π_i, (rounded to 5 decimal places) for encounters within the specified distance range.

Table 7.2 provides a listing of all DTMC steady state probabilities computed from the simulation runs outlined in Chapter 5. This enables us to calculate C_{VDTN} corresponding to each scenario (AP or IBSS, TCP or UDP). It is worth noting two features of the steady state probabilities. The first, rather expected, shows that for larger populations, the probability that two nodes are closer than 200 meters is higher. For a population of 60 vehicles, 12% of the time a vehicle is closer than 200 meters to another one.

The surprising result is that while encounters in the (10,20] interval are rare (and less than for larger intervals), encounters in the (0,10] interval are more likely. The probability that two nodes are within the (0,10] interval is comparable to the two nodes being in the (50,75] interval. A possible explana-

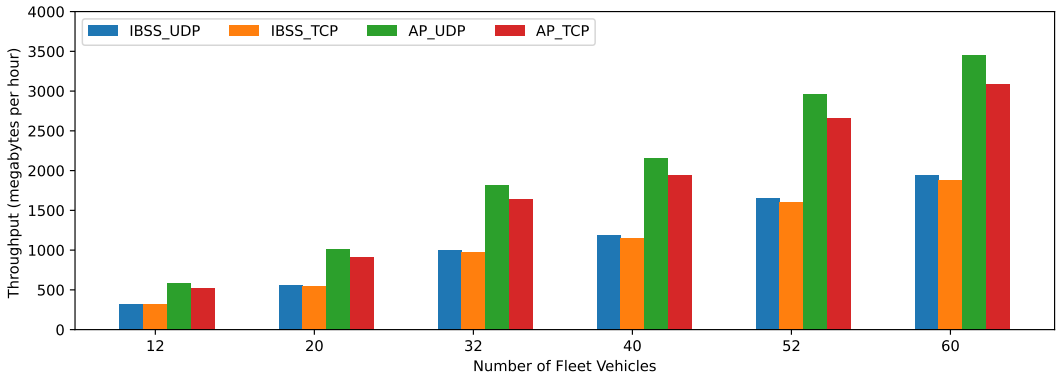


Figure 7.5: C_{VDTN} expressed in Megabytes per hour per vehicle.

tion behind this odd behavior is the impact of traffic lights and intersections. Two vehicles that find themselves within the (0,10] interval are very likely to be within the same interval in the next instant, whenever they are stationary or near-stationary, as e.g., waiting at traffic lights. Vehicles at larger distances are more likely to be in motion (at least one of them) and the distance among them is less likely to remain within the same interval at the next time step. Moreover, the (0,10] interval is not as uncommon, happening at about 0.9% of the time when the population is 60 vehicles.

The combination of the field measurements and the steady state probabilities provides us the results in Figure 7.5. The absolute numbers are impressive, albeit one has to keep in mind that the field measurements came from an interference-free environment with no challenging wireless propagation features like dense buildings etc. Hence, one should approach the results as optimistic, but informed, estimates. The results in Figure 7.5 assume that $b_i = 0$ beyond D_{max}, while clearly this was not the case. In a future study further distances and concentric intervals may have to be included to do justice to the improved long-range performance adaptation of modern WiFI transceivers.

7.4.1 Quantitative Remarks

We have reached the point where we can assess the VDTN option for our application with some, approximate, quantitative observations.

1. Using the estimated bundle sizes from Section 3.2.2 we can calculate an upper bound for the number of bundles capable of being pushed through the system per vehicle. To simplify things, we assume one-hour bundles of approximate size 1MB each. Hence, the entire fleet produces 60MB per hour for a fleet of 60, and, on the low side, 12MB per hour for a fleet of 12 vehicles. Correspondingly, over the working day of approximately 8 hours, 60 vehicles produce 480MB and the 12 vehicles produce 96MB.

2. Assume that each bundle gets to be, completely, epidemically copied to all other nodes. The total volume of data transfers (minus the originating node) is 59 copies for 60 vehicles, and 11 copies for 12 vehicles. The bulk copying capacity needed is 28.320 GB for 60 vehicle, and 1.012 GB for 12 vehicle fleet.

3. Figure 7.5 captures the per-node transfer capacity, but any pairwise communication assumes one node is transmitting and one receiving (half duplex), hence only half of the nodes on average can be making use of the capacity available. That is equivalent to, up to 30 concurrent transmissions for 60 vehicles and up to 6 concurrent transmissions for 12 vehicles.

4. From Figure 7.5, the best (worst) performing option for a population of 60 vehicles and 30 transmitters is 828,648 MB (451,771.2 MB) per day. The corresponding best (worst) option for 12 and 6 transmitters is 27,945.12 MB (15,109 MB) per day.

In summary, daily, there is close to 800 GB leftover capacity in the best case, and close to 423 GB leftover capacity in the worst case for a fleet of 60 vehicles to carry additional bundles not originating from the vehicle sensors. Even for a small fleet of 12 vehicles, there is, at best, a 26 GB leftover capacity, and at worst a 14 GB leftover capacity. Hence, the application of a VDTN to collect more data, e.g., from WSN gateways, is feasible while also satisfying the basic requirements of the original vehicular fleet tracking application. While not pursued in this thesis, the inclusion of rest stop AP transfers into the model can only improve the already positive quantitative results. The

very approximate results we have produced also suggest that there is ample "head room" of spare carrying capacity that could be sacrificed in order to absorb the impact of a real environment e.g., wireless interference, non-ideal medium access behavior, routing overheads, etc. yet still be able to support the proportional-to-travel fleet monitoring and management application.

Chapter 8

Conclusions

This thesis is, primarily, a work of synthesis, pulling from multiple areas while remaining faithful to practicality and realism. The thesis describes an attempt to learn about the potential of VDTNs by pursuing the subject on multiple fronts: proof-of-concept design, simulations of large scale deployments, and field measurements. Due to unforeseen circumstances the scope of the field experiments was redesigned, but the simultaneous falling out of favor of the IEEE 802.11p protocol refreshed our interest to look into WiFi as a workable alternative. Interactions with a local SME provided the inspiration for the fleet management application by means of VDTNs.

The emphasis of this thesis is on the traffic volume that a VDTN can carry. An important facet left outside the scope of the thesis is that of VDTN routing latency performance. On one hand, we make a case that epidemic type routing can be appropriate for the kinds of traffic, i.e., vehicle condition information, we consider here, given its relatively low, when compressed, volume needs, and since epidemic routing can minimize the latency among the available options as discussed in the related work chapter. However, one can reasonably argue that if we are ready to accept any delay, why not wait for the end of day when all vehicles return to the depot and can use the local WiFi infrastructure. While this suggestion is valid, we also recognize that the end-of-day data upload may be acceptable, but undesirable, for business process reasons, e.g. if the business staffing is such that no action can take place after working hours. One can claim that via the VDTN transfers, there is a “head start” for essentially free. Another point is that the example chosen can be scaled

to larger areas and fleets operating over large distances, e.g., logging trucks, with fewer opportunities to return to depots and less frequent encounters but more visits to rest stops during a typical working day. In summary, we cannot completely discount the value of delivering the data earlier than when it could inevitably happen – i.e., at the return to a depot or home base.

We did not pay much attention to the impact of rest stop data AP uploads because the results can only get better through their use. The number of locations, frequency of visits, and duration of visits was chosen arbitrarily. Our choices were driven by an intention to capture an absolute minimum notion of a rest stop. That is, locations where the vehicle stops, it is chosen for its proximity to the work route, yet unrelated to the work-related stops. Parking, even if for two minutes, near a rest stop AP can provide upload speeds that are adequate to upload gigabytes of data. However, it would have also been misleading to treat such numbers with some authority since rest stop APs are shared among multiple co-located users, and there may be intentional throttling of the speed in line with them being a "free" service to patrons for which businesses do not wish to pay a higher speed premium.

Our emphasis in V2V communication has been on pairs of vehicles, while clearly there is a possibility that three or more are co-located. From studying the encounter statistics for our simulated scenarios, multi-vehicle encounters were rare, and, of course, more likely for larger fleet configurations. We did not address the performance study of how the airtime is shared among multiple fleet nodes either but a form of "fair" sharing of the encounter time for exchanges may have to be introduced. The results presented in the previous chapter do not consider the case of congestion and interference caused by other WiFi transmissions. As one can readily experience in a city, there are numerous WiFi APs from residences and co-existence with them implies a loss of the pristine channel capacity used by the VDTN.

In absolute numbers the effective number of bits per second that can be, on average, forwarded over a VDTN can be small, especially for small fleets. Nevertheless, those numbers have to also be seen against the backdrop of "free" alternatives such as long range IoT protocols, e.g., LoRa, that achieves at most a few hundreds of kilobits per second.

8.1 Extensions and Future Work

We consider the lack of simulated interference with co-existing WiFi networks as, possibly, the most serious shortcoming of the current evaluation setup. This could be ameliorated, at a high level, with a model of how busy the WiFi channels are for various locations in the city, which would require a form of spectrum use survey. Yet, the utilization of the WiFi channels used at various locations in the city, is also dependent on the time of the day. A fortunate side-effect of this is that, during working hours, locations near residential areas, while they have many WiFi APs, are probably under-utilizing the wireless channel capacity, leaving more of the capacity to be used by the VDTN. Equally, business and school areas may be more congested in terms of WiFi traffic during working hours. Moving to a more thorough consideration of the WiFi capacity availability would require both a spatial as well as a temporal characterization of WiFi use across an entire city.

We also witnessed how a simple scheduling model, like restricting each fleet vehicle to a sector, can result in a wide variance of possible encounter statistics for different vehicles. A single DTMC is supposed to capture a population with statistically identical behaviors. Sector geometry is different from one sector to the next, and it influences the statistics of the vehicles assigned to it, e.g., how far they are on average from each other, etc. It is tempting to, instead, adopt a model with a separate DTMC capturing the dynamics of vehicles for each sector. The challenge such a partition entails is that there are areas at the boundaries of sectors, where nodes of different sectors could encounter each other, and the separation of the DTMCs will be unable to describe those encounters. More elaborate encounter models are an obvious future direction.

An unexpected complication that deserves a thorough survey comes from the realization that commodity network cards/modules, once pushed to operate outside the usual managed (client to an AP) mode of operation, can be severely limited in the capabilities they support, e.g., their ability to become high performance APs. Using existing free device drivers, software, and utilities for Linux can also be limiting on how well the interface capabilities can be utilized. In summary, one cannot develop a VDTN based on WiFi without

a good understanding of the underlying network interface hardware and its capabilities. This is unfortunate both in limiting the options for combinations that can produce a good performance, exploiting the most recent advancements in the IEEE 802.11 standard, and as a matter of principle, since the lower layers of the protocol stack cannot be assumed to be performing equally well (or at all) no matter what interface is used. One realistic low cost extension which provides additional degrees of freedom is to endow each OBU with additional WiFi interfaces, able to sustain communication with multiple peers and on multiple channels.

Bibliography

[1] Abdelkader, T., Naik, K., Nayak, A., Goel, N., Srivastava, V.: A Performance Comparison of Delay-Tolerant Network Routing Protocols. IEEE Network 30(2), 46–53 (Mar 2016), https://doi.org/10.1109/MNET.2016.7437024

[2] Apple: Review Aggregate Throughput for Wi-Fi Networks, https://support.apple.com/en-ca/guide/deployment/depbb01ab2ce/web

[3] Aschenbruck, N., Ernst, R., Gerhards-Padilla, E., Schwamborn, M.: BonnMotion: A Mobility Scenario Generation and Analysis Tool. In: Proc. of the 3rd Intl. ICST Conf. on Simulation Tools and Techniques (SIMUTools). pp. 51:1–51:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (Mar 2010), https://doi.org/10.4108/ICST.SIMUTOOLS2010.8684

[4] Bai, F., Stancil, D.D., Krishnan, H.: Toward Understanding Characteristics of Dedicated Short Range Communications (DSRC) From a Perspective of Vehicular Network Engineers. In: Proc. of the 16th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom). pp. 329–340. ACM (Sep 2010), https://doi.org/10.1145/1859995.1860033

[5] Bentley: LEGION Simulator: Simulation & Modeling Software (Jun 2022), https://www.bentley.com/software/legion/

[6] Bloessl, B., Segata, M., Sommer, C., Dressler, F.: Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype. IEEE Transactions on Mobile Computing 17(5), 1162–1175 (2017), https://doi.org/10.1109/TMC.2017.2751474

[7] Bychkovsky, V., Hull, B., Miu, A., Balakrishnan, H., Madden, S.: A Measurement Study of Vehicular Internet Access Using in Situ Wi-Fi Networks. In: Proc. of the 12th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom). pp. 50–61. ACM (Sep 2006), https://doi.org/10.1145/1161089.1161097

[8] Cailleux, L., Bonatti, C.: Securing Header Fields with S/MIME. Request for Comments RFC 7508, Internet Engineering Task Force (Apr 2015), https://datatracker.ietf.org/doc/rfc7508

[9] Cottingham, D.N., Wassell, I.J., Harle, R.K.: Performance of IEEE 802.11a in Vehicular Contexts. In: Proc. of the 2007 IEEE 65th Vehicular Technology Conference (VTC), 2007-Spring. pp. 854–858 (Apr 2007), https://doi.org/10.1109/VETECS.2007.185

[10] Dede, J., Förster, A., Hernández-Orallo, E., Herrera-Tapia, J., Kuladinithi, K., Kuppusamy, V., Manzoni, P., Muslim, A.b., Udugama, A., Vatandas, Z.: Simulating Opportunistic Networks: Survey and Future Directions. IEEE Communications Surveys Tutorials 20(2), 1547–1573 (2018), https://doi.org/10.1109/COMST.2017.2782182

[11] Demmer, M., Ott, J., Perreault, S.: Delay-Tolerant Networking TCP Convergence-Layer Protocol. Request for Comments RFC 7242, Internet Engineering Task Force (Jun 2014), https://datatracker.ietf.org/doc/rfc7242

[12] eclipse: netconvert, https://github.com/eclipse/sumo/blob/main/docs/web/docs/netconvert.md

[13] eclipse: Simulation of Urban MObility, https://github.com/eclipse/sumo

[14] eclipse: Why Vehicles are Teleporting, https://github.com/eclipse/sumo/blob/main/docs/web/docs/Simulation/Why_Vehicles_are_teleporting.md

[15] Er, N.I., Singh, K.D., Couturier, C., Bonnin, J.M.: Towards A Simple and Efficient VDTN Routing Protocol for Data Collection in

Smart Cities (Jan 2022), https://doi.org/10.48550/arXiv.2108.09044, arXiv:2108.09044 [cs]

[16] FCC: FCC Modernizes 5.9 GHz Band to Improve Wi-Fi and Automotive Safety (Nov 2020), https://www.fcc.gov/document/fcc-modernizes-59-ghz-band-improve-wi-fi-and-automotive-safety

[17] Fietkau, F.: [RFC/RFT] minstrel_ht: new rate control module for 802.11n (Mar 2010), https://lore.kernel.org/linux-wireless/4B8C3A21.2050105@openwrt.org/, (linux-wireless mailing list)

[18] Fishy: Answer to "Intel AX200 ap-mode" (Jun 2021), https://unix.stackexchange.com/a/653994

[19] Gao, S., Lim, A., Bevly, D.: An empirical study of DSRC V2V performance in truck platooning scenarios. Digital Communications and Networks 2(4), 233–244 (Nov 2016), https://doi.org/10.1016/j.dcan.2016.10.003

[20] Grünblatt, R., Guérin-Lassous, I., Simonin, O.: Simulation and Performance Evaluation of the Intel Rate Adaptation Algorithm. In: Proc. of the 22nd Intl. ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM). pp. 27–34. MSWIM '19, ACM (Nov 2019), https://doi.org/10.1145/3345768.3355921

[21] Hackett, S.C.: Model AX200NGW Wireless LAN Adapter Card, FCC ID: PD9AX200NG, IC: 1000M-AX200NG (Feb 2019), https://fccid.io/PD9AX200NG/Attestation-Statements/Passive-Scan-Attestation-4208499.pdf

[22] Hadaller, D., Keshav, S., Brecht, T., Agarwal, S.: Vehicular Opportunistic Communication Under the Microscope. In: Proc. of the 5th Intl. Conf. on Mobile Systems, Applications and Services (MobiSys). pp. 206–219. ACM (Jun 2007), https://doi.org/10.1145/1247660.1247685

[23] IANA: Message Headers, https://www.iana.org/assignments/message-headers/message-headers.xhtml

[24] IEEE: IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments (Jul 2010), https://doi.org/10.1109/IEEESTD.2010.5514475

[25] ISED: Digital Transmission Systems (DTSs), Frequency Hopping Systems (FHSs) and Licence-Exempt Local Area Network (LE-LAN) Devices. Radio Standards Specification RSS-247 Issue 2, Government of Canada (Feb 2017), https://ised-isde.canada.ca/site/spectrum-management-telecommunications/sites/default/files/attachments/2022/rss-247-i2-e.pdf

[26] Jain, S., Fall, K., Patra, R.: Routing in a Delay Tolerant Network. In: Proc. of the 2004 Conf. on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM). pp. 145–158. ACM (Aug 2004), https://doi.org/10.1145/1015467.1015484

[27] Keränen, A., Ott, J., Kärkkäinen, T.: The ONE Simulator for DTN Protocol Evaluation. In: Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques (SIMUTools). pp. 55:1–55:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (Mar 2009), http://doi.org/10.4108/ICST.SIMUTOOLS2009.5674

[28] Ligo, A.K., Peha, J.M., Ferreira, P., Barros, J.: Throughput and Economics of DSRC-Based Internet of Vehicles. IEEE Access 6, 7276–7290 (Mar 2018), https://doi.org/10.1109/ACCESS.2017.2785499

[29] Lindgren, A., Doria, A., Schelén, O.: Probabilistic Routing in Intermittently Connected Networks. In: Proc. of the 1st Intl. Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR). pp. 239–254. Lecture Notes in Computer Science, Springer (Aug 2004), https://doi.org/10.1007/978-3-540-27767-5_24

[30] madwifi: ath_rate/minstrel/minstrel.txt, https://fossies.org/linux/madwifi/ath_rate/minstrel/minstrel.txt

[31] McIlroy, M.D.: A Research UNIX Reader: Annotated Excerpts from the Programmer's Manual, 1971-1986. Tech. Rep. #139, AT&T Bell Laboratories (1987), https://archive.org/details/a_research_unix_reader

[32] Nguyen, T.V., Shailesh, P., Sudhir, B., Kapil, G., Jiang, L., Wu, Z., Malladi, D., Li, J.: A Comparison of Cellular Vehicle-to-Everything and Dedicated Short Range Communication. In: Proc. of the 2017 IEEE Vehicular Networking Conference (VNC). pp. 101–108 (Nov 2017), https://doi.org/10.1109/VNC.2017.8275618

[33] npapanik: Adyton, https://github.com/npapanik/Adyton

[34] nsnam: ns-3 Network Simulator, https://github.com/nsnam

[35] omnetpp: OMNeT++ Discrete Event Simulator, https://github.com/omnetpp/omnetpp

[36] Ott, J., Kutscher, D.: The "Drive-thru" Architecture: WLAN-based Internet Access on the Road. In: Proc. of the 2004 IEEE 59th Vehicular Technology Conference (VTC), 2004-Spring. pp. 2615–2622 (May 2004), https://doi.org/10.1109/VETECS.2004.1391395

[37] Ott, J., Kutscher, D.: Drive-thru Internet: IEEE 802.11b for "Automobile" Users. In: Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). pp. 362–373 (Mar 2004), https://doi.org/10.1109/INFCOM.2004.1354509

[38] Palme, J.: Common Internet Message Headers. Request for Comments RFC 2076, Internet Engineering Task Force (Feb 1997), https://datatracker.ietf.org/doc/rfc2076

[39] Pereira, P.R., Casaca, A., Rodrigues, J.J.P.C., Soares, V.N.G.J., Triay, J., Cervello-Pastor, C.: From Delay-Tolerant Networks to Vehicular Delay-Tolerant Networks. IEEE Communications Surveys Tutorials 14(4), 1166–1182 (2012), https://doi.org/10.1109/SURV.2011.081611.00102

[40] Qiu, H.J.F., Ho, I.W., Tse, C.K., Xie, Y.: A Methodology for Studying 802.11p VANET Broadcasting Performance With Practical Vehicle Distribution. IEEE Transactions on Vehicular Technology 64(10), 4756–4769 (Oct 2015), https://doi.org/10.1109/TVT.2014.2367037

[41] reithmeier: PedSim, https://github.com/reithmeier/PedSim

[42] Rubinstein, M.G., Abdesslem, F.B., Amorim, M.D.D., Cavalcanti, S.R., Alves, R.D.S., Costa, L.H.M.K., Duarte, O.C.M.B., Campista, M.E.M.: Measuring the Capacity of In-car to In-car Vehicular Networks. IEEE Communications Magazine 47(11), 128–136 (Nov 2009), https://doi.org/10.1109/MCOM.2009.5307476

[43] Schaad, J., Ramsdell, B.C., Turner, S.: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. Request for Comments RFC 8551, Internet Engineering Task Force (Apr 2019), https://datatracker.ietf.org/doc/rfc8551

[44] Scott, K., Burleigh, S.C.: Bundle Protocol Specification. Request for Comments RFC 5050, Internet Engineering Task Force (Nov 2007), https://datatracker.ietf.org/doc/rfc5050

[45] Shah, R., Roy, S., Jain, S., Brunette, W.: Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In: Proc. of the 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications. pp. 30–41 (May 2003), https://doi.org/10.1109/SNPA.2003.1203354

[46] Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In: Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN). pp. 252–259 (Aug 2005), https://doi.org/10.1145/1080139.1080143

[47] StatCan: Coaldale AB: 2016 Census, https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/download-telecharger/current-actuelle.cfm?Lang=E&Geo1=CSD&Code1=4802019&Geo2=CD&Code2=4802&B1=Population&type=0&FILETYPE=CSV

[48] StatCan: Coalhurst AB: 2016 Census, https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/download-telecharger/current-actuelle.cfm?Lang=E&Geo1=POPC&Code1=1249&Geo2=PR&Code2=48&B1=Population&type=0&FILETYPE=CSV

[49] StatCan: Lethbridge AB: 2016 Census, https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/download-telecharger/current-actuelle.cfm?Lang=E&Geo1=CSD&Code1=4802012&Geo2=PR&Code2=48&B1=Population&type=0&FILETYPE=CSV

[50] Stevens, M., Nikolaidis, I.: A Flexible On-Board Unit Architecture for Sensor Data and Fleet Management Services. In: Proc. of the 2018 9th Intl. Conf. on Information, Intelligence, Systems and Applications (IISA). pp. 1–8 (Jul 2018), https://doi.org/10.1109/IISA.2018.8633587

[51] Teixeira, F.A., e Silva, V.F., Leoni, J.L., Macedo, D.F., Nogueira, J.M.S.: Vehicular Networks Using the IEEE 802.11p Standard: An Experimental Analysis. Vehicular Communications 1(2), 91–96 (Apr 2014), https://doi.org/10.1016/j.vehcom.2014.04.001

[52] Torgerson, L., Burleigh, S.C., Weiss, H., Hooke, A.J., Fall, K., Cerf, V.G., Scott, K., Durst, R.C.: Delay-Tolerant Networking Architecture. Request for Comments RFC 4838, Internet Engineering Task Force (Apr 2007), https://datatracker.ietf.org/doc/rfc4838

[53] Tornell, S.M., Calafate, C.T., Cano, J., Manzoni, P.: DTN Protocols for Vehicular Networks: An Application Oriented Overview. IEEE Communications Surveys Tutorials 17(2), 868–887 (2015), https://doi.org/10.1109/COMST.2014.2375340

[54] Vahdat, A., Becker, D.: Epidemic Routing for Partially-Connected Ad Hoc Networks. Tech. Rep. CS-2000-06, Duke (Jul 2000), http://issg.cs.duke.edu/epidemic/epidemic.pdf

[55] Vanhatupa, T.: Wi-Fi Capacity Analysis for 802.11ac and 802.11n: Theory & Practice. Tech. rep., ekahau (2015), http://www.ekahau.com/wp-content/uploads/2020/06/Wi-Fi_Capacity_Analysis_WP.pdf

[56] wiki.kernel.org: minstrel, https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel

[57] Wu, J., Lu, H., Xiang, Y.: Measurement and Comparison of Sub-1GHz and IEEE 802.11p in Vehicular Networks. In: Proc. of the 2017 IEEE Symposium on Computers and Communications (ISCC). pp. 1063–1066 (Jul 2017), https://doi.org/10.1109/ISCC.2017.8024666

[58] Xia, D., Hart, J., Fu, Q.: Evaluation of the Minstrel Rate Adaptation Algorithm in IEEE 802.11g WLANs. In: Proc. of the 2013 IEEE Intl. Conf. on Communications (ICC). pp. 2223–2228 (Jun 2013), https://doi.org/10.1109/ICC.2013.6654858

[59] zeromq: The ZeroMQ Project, https://github.com/zeromq

Appendix A

Field Experiment Photos

Figure A.1: ATV with wagon and a power generator.

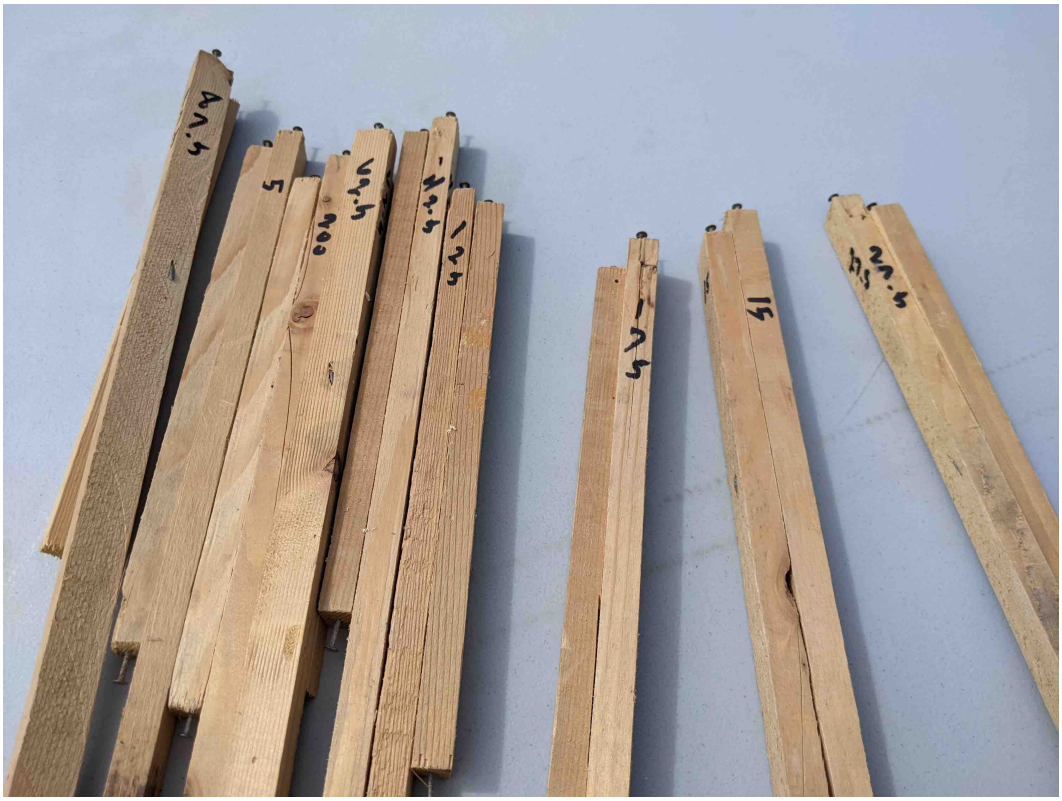


Figure A.2: Stakes used to mark appropriate locations for devices.

Figure A.3: Stake in ground.

Figure A.4: A wheeled measuring tape was used to determine distances between devices.

Figure A.5: Site 1: view from one thin client with the other one off in the background.

Figure A.6: Site 1: view from one thin client with the other one 200 meters in the background.

Figure A.7: Site 2: stationary thin client was positioned at the edge of the backyard (stake closest to the foreground of the picture), with the movable thin client away into the field in the background.

Figure A.8: Site 2: the stationary thin client setup .

Figure A.9: Site 2: the movable thin client setup.

Appendix B

OBU Hardware

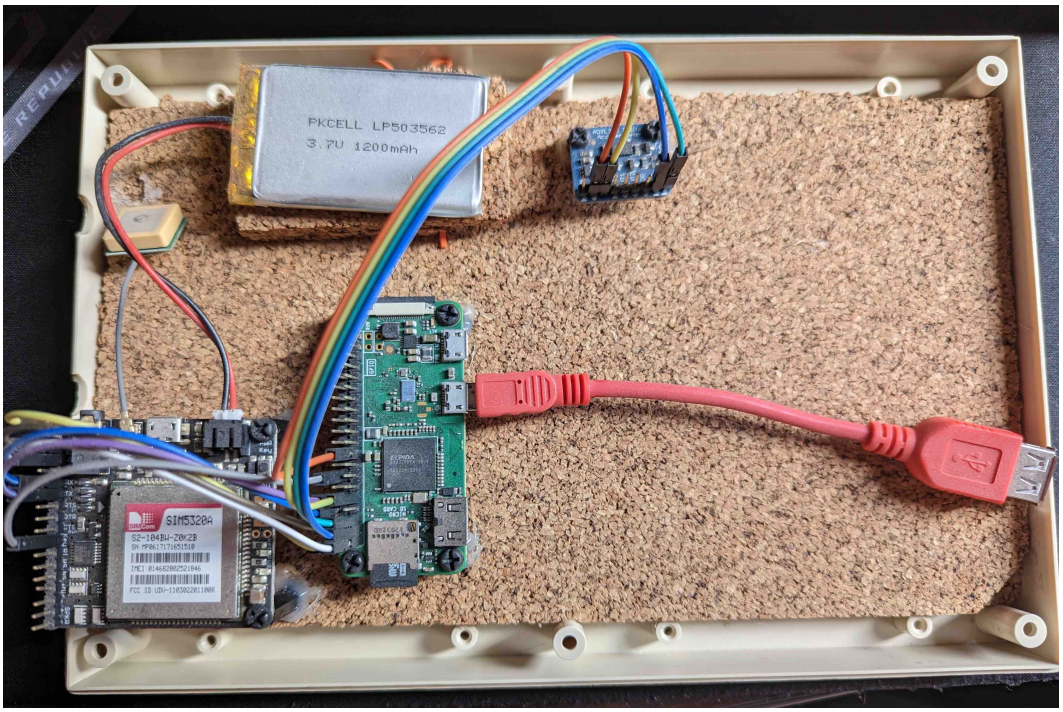


Figure B.1: Prototype iteration 1 (with passive GPS antenna).

The first prototype iteration is shown in Figure B.1. The following can be seen:

- **Bottom left:** GPS/Cellular module. A passive antenna is attached to the top left and a backup battery to the top right of the module.
- **Top right:** Accelerometer.
- **Bottom middle:** RPi Zero W. It connects to the GPS module via a UART serial connection, and the accelerometer module using SDA and SCL

pins via I2C serial communication. The red USB cable was only for debugging purposes while testing. A WiFi USB dongle could be attached, allowing for P2P communication with a laptop.

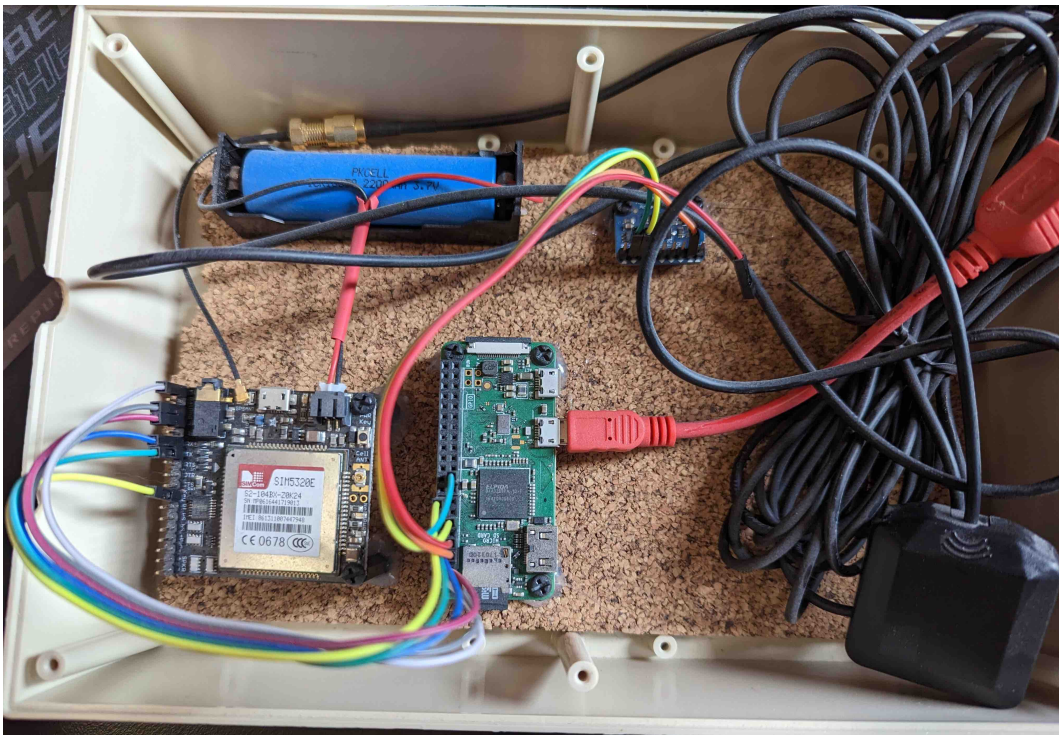


Figure B.2: Prototype iteration 2 (with active GPS antenna).

The second prototype iteration is shown in Figure B.1. The following can be seen:

- **Bottom left:** GPS/Cellular module. An active antenna is attached to the top left and a backup battery to the top right of the module.
- **Top right:** Accelerometer.
- **Bottom middle:** RPi Zero W. It connects to the GPS module via a UART serial connection, and the accelerometer module using SDA and SCL pins via I2C serial communication. The red USB cable was only for debugging purposes while testing. A WiFi USB dongle could be attached, allowing for P2P communication with a laptop.

For access to the vehicle's engine data, a module (Figure B.3) was plugged

Figure B.3: Side view of Bluetooth OBD-II dongle.

into the vehicle's controller area network port, communicating to the RPi via a Bluetooth link.