Imperial College London

Department of Computing

A Framework for Decentralised Vehicular Services

Rudi Ball

December 2011

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London and the Diploma of Imperial College London

Declaration

I herewith certify that all material in this thesis which is not my own work has been properly acknowledged.

Rudi Ball

Abstract

Traffic management is an old and growing problem within cities with inefficient road use resulting in significant economic costs. Existing traffic management solutions are typically large centralised systems which rely on central authorities for control. Furthermore, these systems are costly to setup, deploy and maintain. Within the near future it is expected that Vehicle-to-X (V2X) technologies will become integrated into both vehicles and transportation infrastructures. V2X technologies allow vehicles and road-side infrastructure to communicate with one another using ad-hoc wireless communications.

In this thesis we present a unique *vehicular framework* for the development and prototyping of decentralised vehicular services which exploit available V2X technologies. The framework uses discrete event simulation to evaluate the operation of a decentralised vehicular service. The decentralised services presented within the thesis are unique as they require a combination of scaled ad-hoc *inter-vehicle messaging*, *mobility data* and *cooperation* to enable communities of road vehicles to provide services to one another. Vehicles manage themselves in their local space to approximate the outcomes of centralised services. As vehicular services are decentralised they reduce the costs associated with deploying and supporting traffic services.

Using the framework we prototype two novel decentralised traffic control protocols which evaluate the problems of *travel time estimation* and *intersection control*. Each protocol is evaluated in a scaled scenario which emphasises the usage and requirement of fine grained geographic mobility traces which mimic real city road maps. We show that decentralised approaches provide a feasible means of providing vehicular services to users. We evaluate the trade-offs and performance issues resulting from their use.

Acknowledgements

I am forever grateful and thankful to a large group of people whom I had the privilege of learning from during my time at Imperial College. Foremost, I am indebted to my inspiring supervisors Naranker Dulay and Emil Lupu. Thank you for your gifted motivation, teachings, close support, critical assessment and guidance throughout the years. I am also thankful to a number of other supportive mentors including Morris Sloman, Susan Eisenbach, Peter Pietzuch and Rogerio de Lemos. My great thanks to Amani El-Kholy for her help and advice on administrative issues. My thanks to the Distributed Software Engineering Section and Policy Group with whom I have been associated as well as the Engineering and Physical Science Research Council (EPSRC) and the university for their funding and the tremendous opportunities provided to me as part of both the Cityware and AEDUS2 projects. I am grateful to my thesis examiners, David Hutchison and William Knottenbelt, for their time, comment and examination of the work.

My thanks to my colleagues, past and present: Dalal Alrajeh, Andi Bejleri, Themis Bourdenas, Robert Craven, Driss Choujaa, Domenico Corapi, Luke Dickens, Changyu Dong, Anandha Gopalan, Vaibhav Gowadia, Markus Huebscher, Srdjan Marinovic, Pedro Martins, Edmund Noon, Leonardo Mostarda, Dimosthenis Pediaditakis, Nikos Rizopoulos, Giovanni Russello, Enrico Scalavino, Andrew Smith, Tamas Suto, Daniel Sykes, Vrizlynn Thing, Ryan Wishart and Yanmin Zhu. Thank you for the gift of your time, a myriad of discussions, insights and memories. Finally, a thanks to my family and the Jensen family for their constant encouragement and motivation throughout the years abroad.

"Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse:
complexity sells better."

–

Edsger Dijkstra

'On the Nature of Computing Science'

1984

Contents

List of Tables

List of Figures

1. Introduction

Traffic management is an old, costly and growing issue within road networks. Various control systems have been implemented and applied to traffic management with varying success. An example small scale traffic control system is that of automated traffic lights, which attempt to share the usage of road intersections between competing vehicles. As time has progressed, the declining costs of computer hardware and improvements in computing technologies, have contributed to new control systems being applied to traffic management problems. Some of these systems seek to control traffic, not only within a city but also countrywide. Traffic management solutions have tended towards becoming ever larger and more expensive centralised sensory control systems. However, the existence of cheap sensors, devices and wireless communication technologies offer an alternative decentralised means by which to construct traffic management systems and thereby manage the city at reduced cost.

This thesis presents a *vehicular framework* for the development and evaluation of decentralised vehicular services which are applied to traffic management problems. The framework provides a geographic scaffolding methodology, which helps to distil and abstract away complexities experienced when building decentralised services. Decentralised vehicular services seek to extend the outcomes of centralised services but reduce the costs of setup, deployment and maintenance, enhancing road usage, reducing the costs of road monitoring and management, as well as reducing road infrastructure costs. The services differ in comparison with alternative approaches, as they focus on traffic monitoring and control problems and leverage scaled ad-hoc messaging, fine grained mobility data and cooperation to organically construct vehicular services on demand. In our framework, vehicles act as mobile sensors and as data mules (carrying and disseminating data). As we lack the tools to rapidly prototype and evaluate such services, the framework incorporates discrete event *simulation* as a tool to measure, evaluate and compare service protocols.

1.1. Motivation

Roads provide one of many transportation infrastructures within cities, allowing road vehicles to travel from origin to destination via a set of routes. Road network topologies are typically similar but unique (Figure 1.1). Within a road network, traffic is formed where vehicles conflict with one another in the usage of the road network.

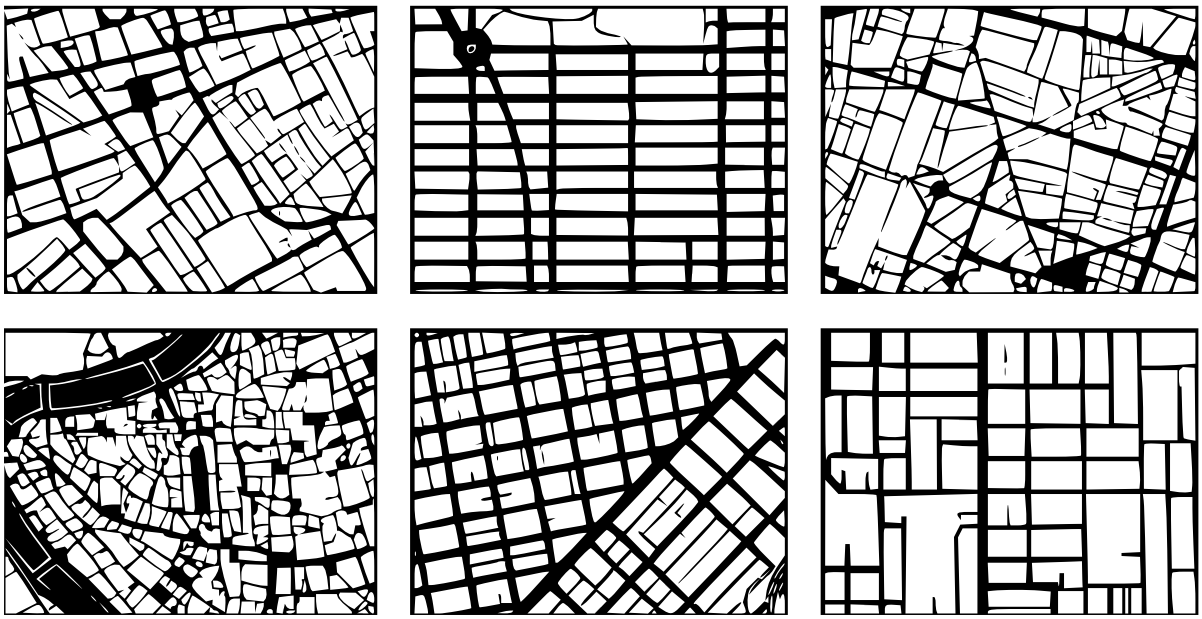


Figure 1.1.: Selection of road network topologies, viewed at the same scale [Jen08].

The underlying expectation from academia [CRK08, LMP+07, LH08, GBT+09, KH95, DS04], government [Par09, BB06] and industry [NBH+11, Bar09, Waz11] (such as the One-Pair EtherNet Alliance Special Interest Group[1]) is that road infrastructures can be monitored and controlled to improve the performance of the city, reducing traffic delays, increasing flows and improving city operations. Management decisions affect the users of the road network (for example vehicles, drivers and pedestrians). Some examples of management methods include the use of road rules, traffic lights and information boards. Road management can either be applied by a *road management authority* (for example traffic light control) or by the *users* of the roads themselves (for example vehicles and drivers deciding on actions such as over-taking or ordering at four-way stops). In each case, it is desirable that management of the road net-

[1] `http://www.opensig.org/`

work should seek to improve the performance of the road network while safely enabling vehicles to journey from origins to destinations.

In general ITS services seek to benefit city users, city planners and city administrators [Baz07]. Some proposed ITS include *safety systems* (warning systems, collision avoidance, virtual traffic signalling), *security systems* (anti-theft and car location applications), and *driver information systems* (maps, smart navigation). Some ITS envision new control techniques and automation using 'smart cities' [RT11, NBH+11] and 'smart cars' [Var93]. Systems are capable of automatically identifying, adapting to and solving traffic problems to optimise the performance of the road network. Thus far, approaches have either used large and costly centralised architectures or isolated road-side infrastructures [BB06, Tho04, Par09]. Success has been mixed and road management remains a costly problem in many cities.

The economic costs associated with a badly performing road network are significant [AS94, PJP09, LH08, GBT+09]. Sub-optimal management may result in travel delays, traffic slow downs, excessive fuel usage, vehicle wear, increased pollution and accidents which in turn translate into monetary costs. Each year thousands of people die in road accidents with the costs of accidents inside the European Union estimated as much as 1% of Gross Domestic Product [Par09]. With traffic management already a problem and city populations expected to grow in size in the future, traffic within the United Kingdom is projected to cost £22 billion per year by 2025. In 2008, 93% of CO_2 emissions in the United Kingdom were attributed to road vehicles resulting in negative environmental effects [Par09]. While the estimated costs associated with road vehicles are large, the costs of managing road networks are also significant. The Department for Transport (DfT) in the United Kingdom estimate the costs of maintaining a centralised road management and payment system to cost between £2-5 billion per year. Given these problems and costs, a great deal of research and development has followed the application of computing technologies to traffic management problems. Broadly these systems are defined as Intelligent Transportation Systems (ITS) [BB06, Sie11] (Figure 1.2).

The proliferation and ubiquity of mobile devices like smartphones has beneficially resulted in the reduced cost of various electronic components. Sensory and communication components like cameras, accelerometers, radios and GPS are available at low cost to device builders and system developers. As such, various devices are now becoming embedded in a wide variety of physical objects. Within the past decade, efforts have been made within the ITS domain to provide

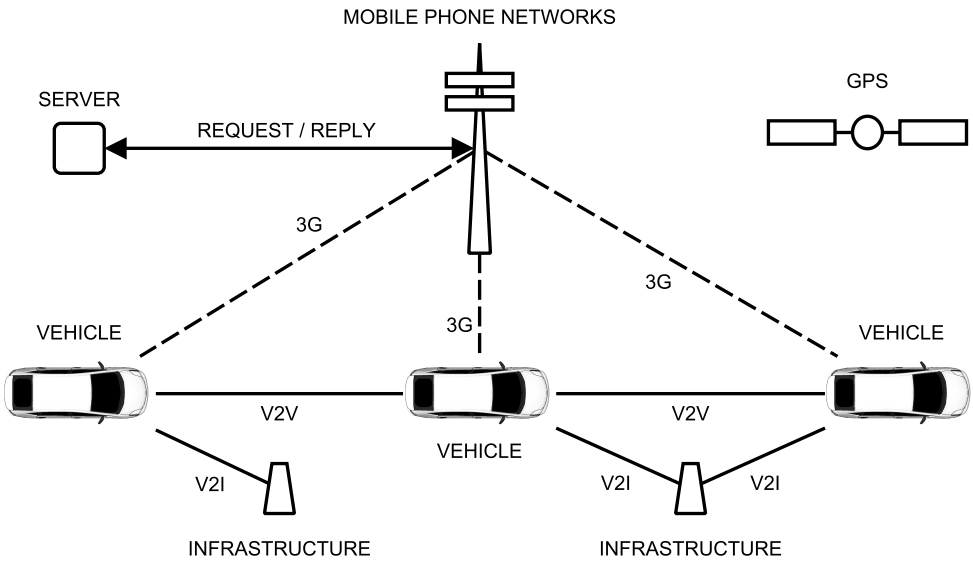


Figure 1.2.: A typical ITS scenario. Multiple vehicles interact with both centralised ITS services via 3G data networks, one another (via V2V interactions) and road-side infrastructure (via V2I interactions). Vehicles and infrastructure use GPS satellites to find position data and calculate speed and orientation data. Decentralised vehicular services operate without service providers (servers) and mobile phone networks.

Vehicle-to-X (V2X) technologies[2] [IEE10, Sie11]. These technologies allow vehicles and road-side infrastructure to communicate with one another. Figure 1.2 depicts some typical V2X sub-scenarios within a more global ITS scenario, where vehicles interact with one another, road-side infrastructures, positioning services (like GPS) and mobile phone networks to provide vehicular services. The combination of V2X and high accuracy positioning presents the opportunity of enhancing traffic services using decentralised approaches (removing service dependence on centralised infrastructure networks such as 3G/4G). If decentralised, previously expensive and costly centralised services can be reduced in cost providing benefit to city users, planners and administrators [Kom10, WER+03, Enk03, LMP+07].

We should be careful to note that centralised systems consider that road networks embed sensors and road-side equipment alongside or within roads. Centralised processing of road data occurs for the management of the road network. Many sensors exist, however a single decision making entity exists to handle these sensory inputs. Decentralised systems distribute the processing of data and management of the road network, ideally close to the producers

[2] V2X are sometimes referred to as Car-to-X (C2X) systems.

and consumers of road data. This could be in vehicles as explored in this thesis, or road-side, or a combination of the two. More generally, decentralised systems have the potential to be more cost-effective, more reliable and more scaleable with many low-end processors in vehicles or roadside versus high-end processors in remote cloud servers. Any estimates of costs need to consider all the costs including hardware and software costs, communication costs, costs of installation and maintenance, costs of compliance to environmental and safety standards, costs of insurance and legal liability. Furthermore costs can be incurred by many parties, infrastructure providers, by those in charge of road planning and maintenance, by vehicle manufacturers, by software/service providers and by drivers.

1.2. Centralised and Decentralised Vehicular Services

Within the thesis we divide vehicular services into: *centralised* and *decentralised* services based on a number of criteria (Table 1.1). Both centralised and decentralised vehicular services assume that vehicles are capable of sensing attributes about their local environment. For example, position, orientation, speed and time. Using positioning (spatial), time (temporal) and memory attributes, a vehicle can heuristically construct more complex abstractions to define both its own state and the state of the immediate environment which it exists within. Current architectures used in traffic management are centralised. Some approaches use cloud infrastructures [Waz11, HWH+10].

	Centralised	*Decentralised*
Messaging	extended (dependent on mobile data network availability)	limited (less than 1 kilometer, dependent on ad-hoc wireless radio broadcast range)
Interactions	small scale, long duration	large scale, short duration and opportunistic
Cooperative	indirectly	directly

Table 1.1.: Service differentiation.

Centralised architectures typically collect, analyse and manage transportation by interpreting vast amounts of data in a centralised management location or authority. An advantage of this is that centralised services are largely available in terms of uptime and simple to develop. Mobile data networks (e.g. 3G/4G) allow vehicles to both access a central service, but also improves the collecting capabilities of the service. Vehicles use a limited number of client-server

interactions to access services (small scale interactions). Data sharing is indirect using the intermediary authority. The predominant disadvantage of their use is geographic cost [Par09]. As road networks are large (with cities measuring tens kilometers in diameter), the scaling of a centralised solution is often prohibitively large. The cost to the service provider is that they maintain a static infrastructure for both the collection and storage of data. Hence, a third party is required to fund and maintain the centralised services. Many centralised approaches require large hierarchies of sensors to collect data. Ideally, traffic management requires that services equally sense all regions of a road network.

Within the V2X domain [LMP+07, WER+03], vehicles and road-side infrastructure cannot broadcast to others beyond their wireless communication range or local proximity. Vehicles use intermittent intervals of contact to disseminate data and services within the road network. Vehicles are assumed, due to their size, to be capable of holding large data stores. Decentralised systems share and cooperate with one another directly, yet only have a subset of available vehicles to cooperate with (those within local communication range). Vehicles require scaled interactions to maximise the chance that data will be disseminated to other vehicles in future contact intervals. V2X technologies, combined with vehicles, allow for decentralisation, such that they remove the requirement for services to use a server and mobile phone network for traffic management (given the ITS domain described in Figure 1.2).

1.3. Challenges

Traffic management within cities is challenging as road networks are dynamic and large. Road network usage and traffic is dynamic, for example, city road traffic may experience certain peak usage in the morning and evenings as people commute to and from home. The physical size of a road network presents challenges to monitoring. In general, road managers monitor road use, analyse this use (using a variety of methods, for example using heuristics, simulation and/or statistical methods) and decide on actions by which to 'improve' the operation of road sections. This management is complicated by the fact that decisions applied to one region of road network influence adjacent regions [LH08, HRBW81, GBT+09]. Assuming that an extensive centralised road management service exists, the results of management decisions may be monitored, analysed and modified more rapidly to adapt to changes in the road network. For

example we might see increasing traffic levels due to either a new traffic source or congestion due to an accident. Ideally, management decisions could better adapt to changes in traffic if centralised systems had more precise data about the intended current or future usage of the road network. For example, if vehicles shared their intended routes with management systems, the road network could measure the number of vehicles intending to use a road section. Traffic lights might be adapted to more efficiently handle these intentions (in reality actuated traffic lights are polled when vehicles drive over sensors in the road, however they lack detail on the future road choices of an individual vehicle).

Decentralised vehicular services developed within the framework are particular as vehicles share data about their past, present and future movement using V2X techniques. Using feedback, the decentralised services continuously adapt service operation to achieve a performance goal. Notably, while a multitude of use cases and applications have been proposed for the V2X setting we lack tools to build, port and evaluate services [LMP+07]. Furthermore, vehicles require means of deriving complex data from positioning (spatial) and time (temporal) primitives, and the decentralisation and scaling of services makes reasoning about them increasingly complicated. Developers are presented with questions on how a decentralised service is set to operate when scaled, i.e. will system performance be similar for increasing numbers of vehicles? What data, when shared achieves the best outcome? How often should we share data? The costs and complexity of deploying a large scale system require that the creators of such systems have confidence of correct operation. There are challenges to collecting, sharing and computing data within the distributed environment. Decentralised data is typically not addressable or immediately accessible. Broadcasting produces redundant data and methods of data management are necessary to manage the effects of redundancy.

There are many other issues that will impact the adoption of vehicular services. Examples include security, privacy, trust, incentives, compliance with safety standards and legal liability. These are important issues but are outside of the scope of thesis. As an example, if we consider legal liability, vehicles are currently required to be driven by a human driver if they are to operate on a road network - the Google car in Nevada is an exception. If an accident were to occur, the drivers involved, might be liable based on the circumstances of the accident. If vehicles do not have drivers then who is responsible if a vehicle is involved in an accident? Many legal obstacles will need to be overcome if driverless vehicles are to be permitted.

1.4. Hypothesis and Assumptions

The primary hypothesis of the thesis is to determine the practicality and performance of designing and using decentralised vehicular services for traffic monitoring and control. The thesis specifically considers decentralised vehicular services that make use of the IEEE 802.11p Wireless Access for Vehicular Environments (WAVE) standard for short messages. Decentralised solutions are compared with centralised ones. The key contribution is the design and implementation of a novel framework for the design and simulation of large-scale decentralised vehicular services. It incorporates a new simulator, Geographic Urban Simulator (GUS) with a number of mobility and communication features that are not available or hard to apply using other simulators, namely (a) control emulation, (b) geographic positioning, (c) dynamic mobility and (d) short messaging. To assess the applicability of the framework, two contrasting traffic monitoring and control services are developed and simulated: a decentralised travel time estimation service and a intersection control service.

The *framework* allows us to develop decentralised vehicular services. In turn, we use the GUS to evaluate the feasibility of services, simulating large-scale Vehicle-to-Vehicle (V2V) scenarios and comparing these results to centralised approaches. We focuses on two traffic management case studies: (a) estimated travel time mapping (Chapter 4) and (b) intersection traffic management (Chapter 5). The work differs from current and previous work in terms of our focus on decentralised V2V services which use adaptive *message broadcast protocols*, *modifiable mobility patterns*, *feedback loops* [AM08] and *scaled interactions* between vehicles to provide vehicular services to one another.

The framework makes several assumptions concerning object and messaging constraints. It assumes that: (a) all vehicles are mobile, (b) all vehicles communicate with one another using Wave Short Messages (WSM), (c) broadcasts occur every 100 milliseconds, (d) vehicles are able to accurately record their position, and (e) the mobility of a vehicle is constrained to a road network. In this thesis we consider vehicle only architectures, where all processing is done by vehicles i.e. there is no road-side infrastructure for sensing, processing or control. More complex architectures that use vehicles, road-side and cloud infrastructures are also feasible and may lead to better performance and other properties. We believe that our framework is able to model and simulate services for such architectures, but a detailed evaluation requires a

much longer time-frame and reconsideration of the overall design and evaluation methodology needed for mixed mobile and static infrastructures spanning city-wide locations.

1.5. Contributions

We make the following significant contributions:

Framework: a novel framework for the development of decentralised, mobility sharing and feedback-driven vehicular services (Chapter 3). The framework provides a scaffolding for service construction. It allows us to abstract complex heuristics and distil service development while focusing attention on protocol design and traffic management problems;

Dynamic Mobility Specification: a simplified mobility specification for use in modelling and specifying vehicle movements at run-time (Section 3.2.3). The mobility specification considers mobility in terms of geographic mobility *plans* which may be modified during the operation of a service. Mobility patterns are not limited to just historic patterns but include the capability of specifying future expected mobility (known as future tracks). Patterns can be synthesised or converted from real trace datasets;

Geodetic Simulator: a new geodetic simulation environment, the GUS is used for the simulation and evaluation of decentralised services on real city road networks (Chapter 3). The simulation framework emulates the geographic positioning interfaces available to protocols and services. The simulator allows us to build and simulate services using Java, such that protocols may be assessed and directly ported to mobile device platforms like Android[3]. Evaluation within the simulator provides confidence in the services developed prior to deployment testing;

Travel Time Estimation: a novel decentralised travel time estimation service and protocol (Chapter 4) is successfully demonstrated, highlighting the feasibility and trade-offs of constructing decentralised traffic mapping services. Vehicles map travel times using geographic positioning and time data. Travel time tuples (fragments of mobility) are shared with neighbouring vehicles to provide vehicles with dynamic travel time maps, which may be used to dynamically re-route vehicles to allocate traffic to underutilised roads. We

[3] `http://www.android.com/`

measure the performance of the approach in terms of messaging, data recency and data availability;

Intersection Control: a novel decentralised intersection control service (Chapter 5) is developed and compared in relation to existing intersection control approaches. Vehicles approaching an intersection share mobility fragments with one another to order themselves through a road intersection without colliding. We find that, while decentralisation is feasible and potential benefits exist, rapid adaptation requirements make the service difficult for use with human drivers. Rather driver-less vehicles may be better suited to such services [Thr10, Thi11].

1.6. Statement of Originality and Publications

This thesis describes work carried out in the Department of Computing at Imperial College London between 2008 and 2011. I declare that the work presented in this thesis is my own, except where acknowledged. During this period the following related publications have been authored in collaboration with a number of my colleagues:

Rudi Ball and Naranker Dulay. Enhancing Traffic Intersection Control with Intelligent Objects. First International Workshop the Urban Internet of Things 2010: Programming the Real-Time City. Tokyo, Japan. December 2010 [BD10].

Leonardo Mostarda, Rudi Ball, Naranker Dulay. Distributed Fault Tolerant Controllers. DAIS 2010: 141-154. May 2010 [MBD10].

Enrico Scalavino, Giovanni Russello, Rudi Ball, Vaibhav Gowadia, Emil Lupu. An Opportunistic Authority Evaluation Scheme for Data Security in Crisis Management Scenarios. 5th ACM Symposium on Information, Computer and Communications Security. April 2010 [SRB+10].

Rudi Ball, Naranker Dulay. Approximating Travel Times using Opportunistic Networking. 2nd IEEE Intl Workshop on Opportunistic Networking. April, 2009 [BD09].

1.7. Thesis Structure

The remainder of the thesis is presented as follows:

Chapter 2 is concerned with describing the state of the art in traffic management. We explore past and current approaches as well as the limitations of existing methods and models. We discuss new wireless technology standards applicable to the transport domain and works in Vehicular Ad-hoc Networks (VANETs), Multi-agent Systems (MAS), Intelligent Transportation Systems (ITS) and Geographic Information Systems (GIS) which seek to find solutions to specific traffic management problems.

In Chapter 3 we discuss and motivate our vehicular framework, a means by which we may construct decentralised vehicular services. Services developed with the framework use shared and fragmented mobility data (both past and future), feedback loops, heuristics, sensory data and actuation for their provision. We describe a software architecture in which a protocol and application are provided to a driver in an advisory capacity. The second portion of the chapter is concerned with the implementation of the framework known as the Geographic Urban Simulator (GUS) on top of a discrete event simulator known as Java in Simulation Time (JiST). The combination of JiST and GUS allows us to develop decentralised traffic management services in Java. Furthermore, we simulate services prior to deployment such that we may assess and estimate future service behaviour within traffic scenarios. GUS makes decentralised vehicular service development highly accessible. We briefly compare the performance of the GUS with alternative simulators.

In Chapters 4 and 5 we present two decentralised vehicular services: (a) travel time estimation and (b) intersection control. For each service we develop and present a single protocol, which has been constructed using the framework from Chapter 3. Each solution is implemented in Java and evaluated using discrete event simulation methods. For each service we assess the suitability of the solutions and evaluate the assumptions, limitations and trade-offs of the approaches in comparison with existing traffic management approaches.

In Chapter 6 we conclude by summarising the contributions of the work. We also discuss future research directions.

2. Background

In this chapter we describe and evaluate related work, concepts and technologies for vehicular services. As traffic management is an old problem, several approaches have attempted to improve the efficiency and control of road networks, with varying success. We consider the successes, failures and trade-offs exhibited by these systems, as well as their architectures and methodologies.

2.1. Complex Systems

Cities have been viewed as complex systems with large numbers of road users (both pedestrians and vehicles) [Bat07, Baz07]. Road users and road infrastructure influence one another and hence represent a large number of interconnected components or entities. Modelling large scale complex systems is difficult and computationally intensive requiring methods to optimally model individual elements within a system. For example, a modelled vehicle may consist of hundreds of interacting components. In a homogeneous vehicle simulation, these components are replicated for every vehicle existing within the road system. The addition and emulation of other systems, such as messaging and physical phenomena, interactions between vehicles, increase the complexity of such systems yet further.

When distributed components within the system interact with one another in simple behaviours they lead to complex behaviours [GM57, Bat07]. A behaviour which evolves over time due to large scale interactions is known as an 'emergent behaviour' [Lew74]. Emergent behaviours are sometimes counter-intuitive and unexpected. A variety of emergent behaviours are associated with processes in biology, chemistry and physics. For example flocking behaviours have been deemed an emergent behaviour of birds and fish [Rey87]. Emergent behaviours may be negative, however various control and feedback driven processes have benefited

from emergent behaviours including [VVPV97, PDAV08]. Understanding system behaviours is paramount within the context of complex services. An awareness of emergent behaviour is necessary as it guards against unintended effects and side effects produced by specific design decisions and patterns [Ste90].

2.2. Vehicles

Present and future vehicular services rely on the availability of a number of enabling technologies and vehicular capabilities. Vehicles represent atomic communicating components in vehicular systems. Some technologies to enable such systems are already widespread and embedded within mobile devices like smartphones. Others are expected to become more widely used and available within the coming decade. As vehicles are mobile, we focus on mobile device capabilities and wireless radio communications standards. These capabilities influence the modelling and design considerations made when considering complex vehicular services.

2.2.1. Computational Devices

Vehicular devices can be split into two groups, namely: (a) *personal mobile devices* and (b) *onboard vehicle-based devices*. Personal mobile devices are separable from a vehicle, while onboard computers are embedded within a vehicle. This presents subclasses of devices which differ in their capabilities. Both groups are capable of storing gigabytes of data, more than sufficient for many applications. Personal mobile devices embed a variety of sensors including accelerometers, digital compasses, decibel meters, cameras, light sensors and Global Positioning System (GPS) chip-sets. Wireless technologies like Bluetooth (IEEE 802.15x) and WiFi (IEEE 802.11a/b/g/n) can also be used to sense data about the local area in which a device resides. Mobile phone network cell information allows for granular positioning [Zan09], while GPS typically provides more accurate positioning.

Onboard computing technologies were initially used as diagnostic computers within modern vehicles. With access to hundreds of on-board sensors (accessing data such as tyre pressure, engine mixture ratios, brake wear measurements, etc.) These onboard devices collect, analyse and monitor the health status of a vehicle - automatically alerting the driver to maintenance problems. As technology has progressed, the onboard computer has also become responsible

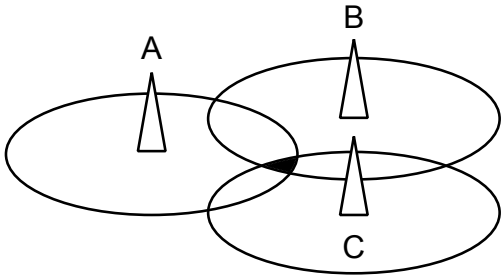


Figure 2.1.: Trilateration scenario: a vehicle determines its location to lie in the intersection of three mobile cell tower signals (A ∩ B ∩ C).

for in-car services such as navigation, communications and safety. Platforms like Android[1] are helping to combine group capabilities. Notably, embedded vehicular computers have access to more low level vehicle data, while personal devices are limited to their own mobile sensors. Both groups are assumed to use the vehicle electrical system as a power source implying that devices have a limitless source of energy. This is a major differentiating factor between Mobile Ad-hoc Networks (MANETs) and Vehicular Ad-hoc Networks (VANETs), where MANETs typically have limited stores of electrical power in the form of cells.

2.2.2. Position and Mobility

For V2X approaches, accurate positioning is extremely important as the distances and tolerances between moving vehicles are typically small. Collating position and time data we can infer and calculate a number of other attributes about a vehicle, such as speed, bearing (orientation) and predicted track [PS96]. Furthermore, an error is associated with each position calculation. Using geographic positioning we can associate an object point on the irregular surface of the Earth (using Geodetic standards). Maps and sets of optimisation algorithms are capable of providing extra contextual information. To improve positioning accuracy, approaches like Differential GPS (DGPS) have been developed as an enhancement (improve accuracy below the 5 meter range [PS96]).

Where GPS is not available, it is possible to find a vehicle's position with increased granularity using the trilateration of mobile phone network cell data (Figure 2.1). However, the error is often no better than a few hundred meters. Similarly, vehicles can use their proximity

[1] `http://www.android.com`

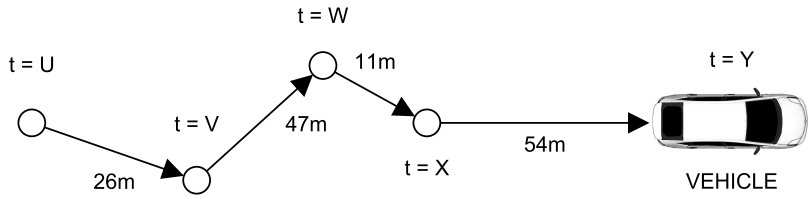


Figure 2.2.: Example mobility trace (history) shows a vehicles movement between a set of geographic positions (circles). The time (t) when a vehicle existed at a location or sampled its position is shown by the set of times between U and Y.

to known road infrastructure to calculate position. For example, some services, using collected databases of WiFi router positions (e.g. Skyhook), provide a geographic position using the overlap, proximity and signal strength of WiFi signals [CCC+06, Kaw09, Zan09]. Proximity positioning using Bluetooth, WiFi and cell information is often favourable in situations where a GPS signal is either weak or not available; however it is typically more granular. Position accuracy with all wireless systems is affected by a number of conditions including weather conditions, radio signal reflection and interference, speed and network availability.

Where the position of a vehicle is observed over a sequence of time instances, vehicles are said to possess a *mobility* [CBD02]. For example, a vehicle driving a route from origin to destination would be constrained to produce a conforming mobility pattern to match the road network where such patterns are recorded as *trace* datasets (Figure 2.2). Recorded mobility datasets are known as 'real traces' while traces generated from algorithms and heuristics for simulation are known as 'synthetic traces'.

2.2.3. Vehicular Communications

If considering individual inter-vehicle communications or links, communications between vehicles and fixed infrastructures can be split into either (a) *long-range* and (b) *short-range* communications. As specified by the IEEE 802.11p standard, short-range communications consider techniques limited to 1000 meters, while long-range communications consider techniques beyond 1000 meters. The rapid adoption of mobile devices has seen the creation of large mobile cellular phone networks to provide coverage for service delivery. The principle basis behind these large and costly networks is to provide a centralised connective infrastructure. The integration of mobile phone technologies into vehicles allows vehicles to communicate with

one another over long-range distances (as far as mobile phone network coverage allows). Both third and fourth generation data networks (3G and 4G) provide an assortment of techniques for data to flow between vehicles and fixed infrastructure (e.g. HSPA/WCDMA and quad-band GSM/GPRS/EDGE), although no simple standard exists for data flow. In situations where mobile coverage is poor these services cannot be used (e.g. inside tunnels). Mobile network data is generally slow in comparison to more immediate methods of proximity communication such as ad-hoc WiFi.

Since 2001 initiatives by the motor vehicle industry have sought to allow vehicles to communicate with one another directly using ad-hoc means for the provision of collision avoidance and safety systems [For11, HFI+07, Sie11, JD08]. A new V2X derivative standard was proposed to support reliable broadcast communications at high speed (where connections could be established at relative object velocities of up to 500 kph), longer range communications, in the presence of multipath reflections and operate with overlapping ad-hoc networks. An extended version of the WiFi IEEE 802.11a radio standard was derived in the form of IEEE 802.11p for V2X scenarios. The standard has been included in the Dedicated Short Range Communications (DSRC) and Wireless Access in Vehicular Environments (WAVE) standards family [JD08, IEE10].

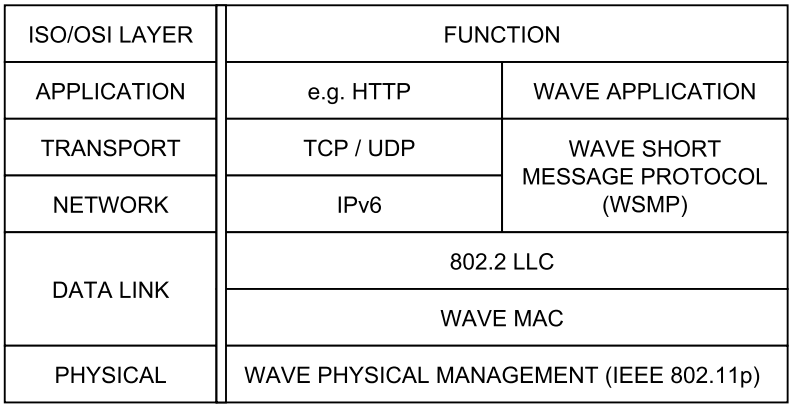


Figure 2.3.: Wireless Access for Vehicular Environments (WAVE) stack.

WAVE operates in the 5.85 to 5.925 GHz range, uses a half-duplex radio link, considers a minimum 200 meter (maximum 1000 meters) communication range and a 6 to 27 Mbps throughput. The standard, being half-duplex, does not allow concurrent send and receive communications. Each communications cycle occurs every 100 milliseconds. Within the OSI stack

(Figure 2.3), WAVE handles the application, presentation, session, transport and networking layers, while DSRC handles the data link and physical layers. Incorporated WAVE components provide a resource manager and messaging, networking (addressing, routing and WAVE Short Messaging) and security services (secure message formatting). IEEE 802.11p supports two different packet transmission stacks, namely IPv6 [Int08] using service channels and the WAVE Short Message Protocol (WSMP) which allows services at the application layer to control radio channels and transmission power. With vehicles having large reserves of electrical power and fuel, wireless communications do not place large demands on the total energy usage of the vehicle. While a number of alternative communications methods exist, including Bluetooth [FP05], ZigBee [BPC+07] and WiMAX (Worldwide Interoperability for Microwave Access, IEEE 802.16) [PH09], WAVE is specifically tailored for V2X scenarios.

WAVE provides us with a secondary method of communicating between vehicles using a simplified connectionless paradigm, namely short wireless broadcasts. While connection-orientated overlays could be constructed from connectionless components, broadcast based store-and-forward approaches are useful in contexts where data is broadcast rather than unicast or multicast to specific destinations. In comparison to IP approaches, communication discovery and setup times do not exist with WSMP. A trade-off of using broadcasts to disseminate data is that data management is fundamental to WSMP operation. Broadcasting favours applications where all vehicles may use a particular data stream. For example, all vehicles may be interested in the weather readings within a particular region. Epidemic routing presents itself as a method to copy data to all vehicles within a system.

While a variety of wireless communications standards exist we focus on IEEE 802.11p messaging. Message management can be split according to the OSI layer (Figure 2.3), where the stack provides UDP/TCP/IPv6 provisions separately to WAVE Short Messaging (WSM). Thus far, VANET approaches have largely considered network layer UDP/TCP/IPv6 and not WSM capabilities.

2.3. Vehicular Ad-hoc Networks

Where multiple vehicles interact with one another using ad-hoc communication, they form Vehicular Ad-hoc Networks (VANETs). VANETs represent the dynamic wireless networks

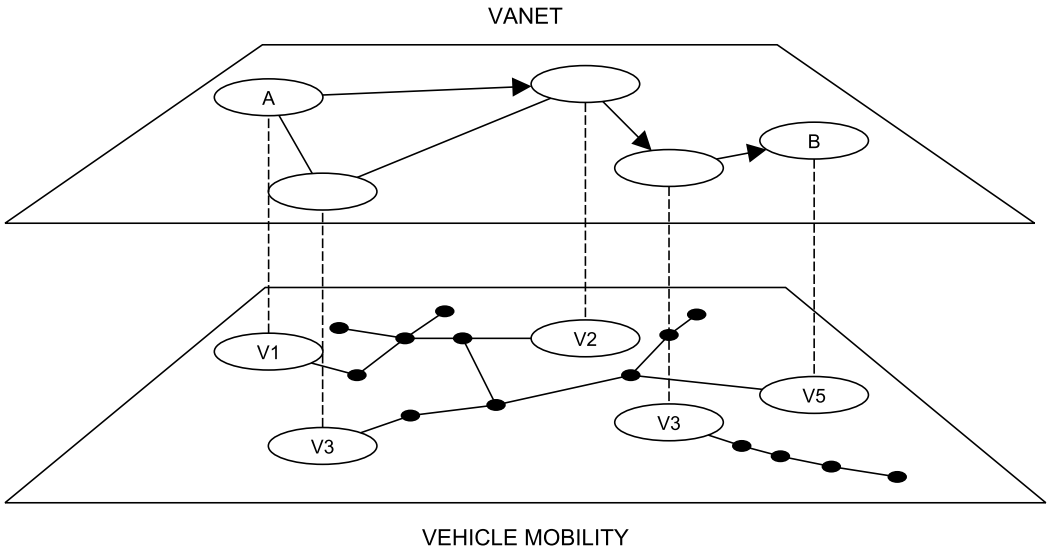


Figure 2.4.: Overlay network where messages are routed within a VANET from A to B.

formed by vehicles as they move along a road network over time [LW07]. Multiple ad-hoc links give the illusion of a permanently connected network over time. Each vehicle operates as both a router of data when connected and as a data mule when disconnected (carrying data from one location to another). This assumes that vehicles are addressable. Layers construct overlay abstractions to simplify systems (Figure 2.4). As vehicles are mobile and wireless ad-hoc communication range is limited. Such networks are normally 'best effort'. In this sense, VANETs attempt to construct a network given the limited ad-hoc connectivity between vehicles.

A large variety of Vehicular Ad-hoc Networking (VANET) works have investigated approaches to routing packets using overlays (Figure 2.4) above intermittently connected ad-hoc wireless networks; some include [MGL04, LHT+03, MWH01, PGHC99, MM09, LJC+00, VB00, LHT+03, SPR05, LM07, BCSW98, ABFeH07, BGJL06, BMJ+98]. Specifically we focus on a subset of routing approaches.

Flooding approaches represents the simplest method of message communication within a VANET. As no message store is assumed to be held by each vehicle, each message received by a listening vehicle is instantly re-broadcast. Flooding achieves a minimal end-to-end delay in message dissemination (i.e. best performance). A trade-off of this performance is that flooding systems suffer from maximum message overheads, as well as cyclic copying.

Flooding serves as a method to determine the minimum end-to-end messaging delay as well as the maximum overheads expected for a given scenario.

Gossip or epidemic routing approaches use a memory store to record retrieved messages. At a later time the vehicle re-transmits the store to neighbouring vehicles. Gossip and epidemic style dissemination while following a similar approach as that of flooding [VB00], notably store and carry data. They are hence considered store-and-forward approaches. The dissemination of data, as in flooding, resembles the infection of a virus in a biological system. Assuming a connected non-mobile network, if every node forwards a message propagation requires $O(1) + log_2 n + \ln n$ rounds to reach all n nodes [DGH+87]. Gossip protocols are differentiated by their management of retrieved data and design properties which decide when and where to broadcast previously retrieved messages. For example, Self-Limiting Epidemic Forwarding proposed by El Fawal et al. [FyLBS+06] attempts to limit the negative effects of message redundancy by applying adaptive ageing, limiting the forwarding factor and controlling message broadcast rates. Other protocols use time events to decide when to broadcast data [SPR05], some use neighbour counts, some broadcast based on acknowledgement of message retrieval, some the type of the receiver [PGHC99], some aggregate data retrieved, some depend on the content of a message (content-based approaches) [MM09] and some use positioning information, geographic measurements or a knowledge of mobility to determine broadcast and retrieval [BCSW98, LM07, MGL04, MWH01, DSW06, LJC+00].

Gossip protocols are in effect a form of controlled flooding which attempt to optimise the performance of flooding and reduce the overheads. Similar strategies and routing algorithms used within Delay Tolerant Networks (DTNs) and Opportunistic Networks (ONs) are sometimes applied to VANETs. DTN works, initially proposed by Cerf [CBH+01], sought to address challenges pertaining to messaging in 'heterogeneous' networks where networks were disconnected for extended periods of time - an issue common with the operation of an inter-planetary network. DTN work was initially motivated to integrate disrupted networks where disruption occurred due to message failure, fault, the unavailable routers or limited power [Fal03, BHT+03]. Messages are 'bundled' together to improve messaging performance. DTNs use 'virtual' network infrastructures to route data. In contrast, ONs approaches like Haggle [SHCD06], do not

possess a known or indexed infrastructure. ONs (sometimes referred to as Pocket Switched Networks) consider a similar context to DTNs however, routing devices are mobile and networks, dynamic and sparse.

2.4. Control Systems

Control systems have long been used to manage, command or regulate the actions of components within various applications (e.g. control units and industrial process control systems). Control systems process sets of inputs and compose sets of outputs based on rules (distilled as if-then-else rules). Within the vehicular environment control systems are used to manage sensing, processing and actuation. There are many approaches to applying control - we illustrate the differences between Feedback Control Systems (FCS) [AM08] and Multi-Agent Systems (MAS). Both approaches have been used or proposed for the provision of ITS services.

2.4.1. Feedback Control Systems

Within engineering, FCS are a practical method to solve problems where optimal operation is known to exist within a bounded region of measurable performance [AM08]. Where control systems use a series of functions to process inputs and return outputs, feedback allows a function to re-evaluate a previous output in a future time step. This provides the function a means of comparing an output in a previous time step with the output computed in the present time step. Both positive and negative feedback are possible.

Systems use feedback loops to adapt control based on present and past inputs [Bak09]. Applied controls are either continuous (where actions manage the system continuously) or discrete (where controls are applied at specific time instance). In effect, feedback allows a controller to be self-correcting and self-regulating. For this reason, FCS are commonly used to deal with dynamic situations or scenarios, where controllers need to adapt to measured errors and maintain the operation of a system within a threshold band of operational performance.

FCS is a well developed area of engineering and commonly associated with industrial systems. As such, FCS have been present on vehicles for a long time in both analogue and digital forms. An example feedback control system is cruise control. Cruise control feedback systems attempt to maintain a specific speed within a bound. For instance, if a vehicle is set to cruise at 40

mph a vehicle attaining 35 mph will automatically speed up or readjust its speed to match the desired cruising speed. Control systems have been implemented as a set of concurrent processes which monitor and respond to changes in an environment. A requirement of many control systems is their operation as real-time software. Real-time assumes that such systems are capable of reacting appropriately to changes in the environment - a clock is added within the control. Timely response is often required due to the nature and applications where these systems are applied.

2.4.2. Multi-agent Systems

Multi-agent Systems (MAS) are characterised as systems of multiple interacting 'intelligent' agents [SV00]. Agents are partially autonomous and decentralised. Each agent holds a local view of its environment. Communities of agents are often termed 'agent societies'. Simple reflexive agents are themselves an evolved form of control system architecture. Agents take as input a series of 'percepts' (perceptions) from an environment and sensors interpret percepts to produce a state view of the world. A set of condition-action-rules are subsequently used to interpret appropriate actions via actuators, which in turn effect the environment. Reflexive agents highlight a single feedback loop. The looping action is normally continuous and non-terminating.

MAS offer a method by which designers can improve system efficiency (via asynchronous parallelism), robustness and reliability (failed agents can be replaced by redundantly available agents), reduce costs (simpler components reduce the cost and complexity of systems) and allow for scalability (the system can be scaled to solve larger problems without reducing performance) [Wei99]. Agents are usually homogeneous and replicated. Specifically MAS attempt cooperation in distributed problem solving and coordination.

2.5. Simulation

As interest in V2V systems has grown so has the need for tools and methods to measure the feasibility of approaches. Few real V2V systems exist. Experiments using vehicles are often expensive, difficult to setup and in the case of complex scenarios measurements sometimes non-repeatable. Large corporations have typically been the only groups able to afford

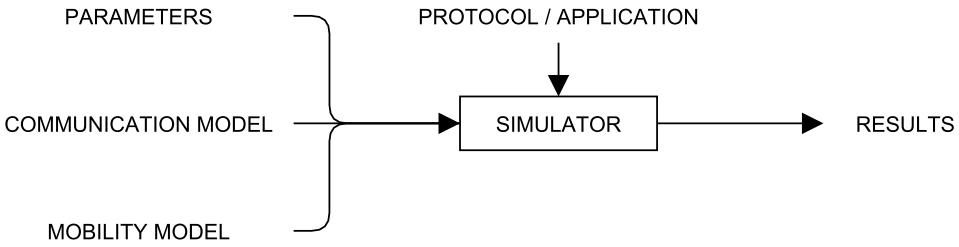


Figure 2.5.: Simulation: frameworks provide a simulator with mobility models, communication models, protocols or applications and scenario parameters. We assume that the simulator holds its own internal object model which includes vehicular models.

both simulation and actual vehicles to robustly evaluate system and application performance [For11, Thr10]. As a consequence most approaches have depended on simulation rather than analytical means as a way of testing the feasibility of protocols and applications. To simulate and experiment with vehicular scenarios many approach vehicular simulation by separating vehicle mobility and messaging simulation. Figure 2.5 illustrates this approach.

Within the scope of V2V simulators, the majority of simulators available are VANET simulators. The simulators focus on network layer routing and not specifically protocol and application (service) construction. Furthermore, no single framework considers all micro and macro simulation features and many differ considerably from one another in terms of the metrics used to measure protocol and application performance.

Simulator architectures typically follow a similar process. Multiple individual vehicle mobilities are modelled. The combination of mobilities produces a collection of vehicular scenarios. A framework is used to superimpose applications above these mobility and communications layers. Metrics and traces are used to measure protocol or application performance. As such simulators require at least three constituents: (a) mobility models, (b) methods to model adhoc communication networks and (c) protocol logic. A set of scenario parameters are used to consider variations in vehicle capabilities and protocol settings.

Simulation trade-offs exist in terms of realism, computation, time and accuracy. Typically, as simulators attempt to provide a more realistic simulation more features are added to the simulator. Depending on the implementation, more features can result in more 'accuracy' in estimating how a protocol or application may behave if applied to real vehicular networks. However, more features require more time and computing resources when simulating a scenario.

Given these trade-offs, it is recommended that the protocol or application designer identify those features which are considered most relevant to a specific application and most useful in answering the specific question being posed about a system. While a number of VANET simulators exist [GC08, MI04, ABFeH07, MWS+05, Cha99, KOK09, Sch88], many simulators consider just the communications models for packet switching and do not include methods by which to stipulate unique vehicle parameters or mobility patterns.

2.5.1. Mobility Models

The intention of using traces is that they 'realistically' describe the movement of vehicles in space for a given scenario. As building and deploying real services is costly, overlay network protocols and applications are typically simulated in combination with traces to gauge service feasibility using various metrics. As V2X services are ad-hoc, service performance is directly affected by mobility. A problem with traces is that many are scenario specific and consider small vehicle populations and only a few interactions between objects over short durations. Furthermore, accurate traces are costly to acquire and privacy concerns make release of some trace data a legal problem for organisations. For the purposes of experimentation, accurate large scale and long duration realistic traces are rare or not available publicly. This is partly because mobility trace datasets have commercial value.

However, a number of scenario specific and open vehicular networking trace data sets exist [FGH+06, Mah07, JHP+03, EP05]. The DieselNet initiative has provided the GENI test-bed and a number of traces [BGJL06]. The project used 35 DieselNet buses containing 'Diesel Brick' computers. Bricks provided DHCP WiFi access to passengers and pedestrians in proximity of buses. As time has progressed, it has become easier for commercial companies to collect such datasets given services such as Google Latitude [Bar09] and Waze [Waz11].

A lack of real traces has motivated the generation of *synthetic* mobility traces. These traces represent movement patterns based on sets of algorithms. The algorithms attempt to 'realistically' generate mobility traces [HFB09]. As synthetic traces are based on algorithms, there is often a concern that such traces are not realistic enough to effectively measure the performance of the system. Vehicles, while constrained to travel along road networks, are affected by a multitude of conditions (both internal and external to a vehicle). Fine grained mobility traces may be required to mimic second by second movement, while coarser grained methods may consider

larger time intervals between object positions. Several works highlight the effect which mobility traces have had on the performance of VANET protocols [CBD02, Bet01a, YLN03].

Various methods have been used to synthesise VANET traces. More recent traffic simulation models have used 'macro' and 'micro' simulation [Hel01]. Macro-mobility simulations attempt to model influences such as road topology, constraining speed limits, lanes, overtaking, safety rules, traffic signs, weather conditions etc. which effect a vehicle. Micro-mobility simulations attempt to simulate driver behaviours and behaviours between interacting drivers, 'driving attitude', including a number of commercially licensed traffic simulators exist for traffic management and analysis in city planning such as PARAMICS [CD96], CORSIM [OZRM00] and TRANSIMS [NBB99].

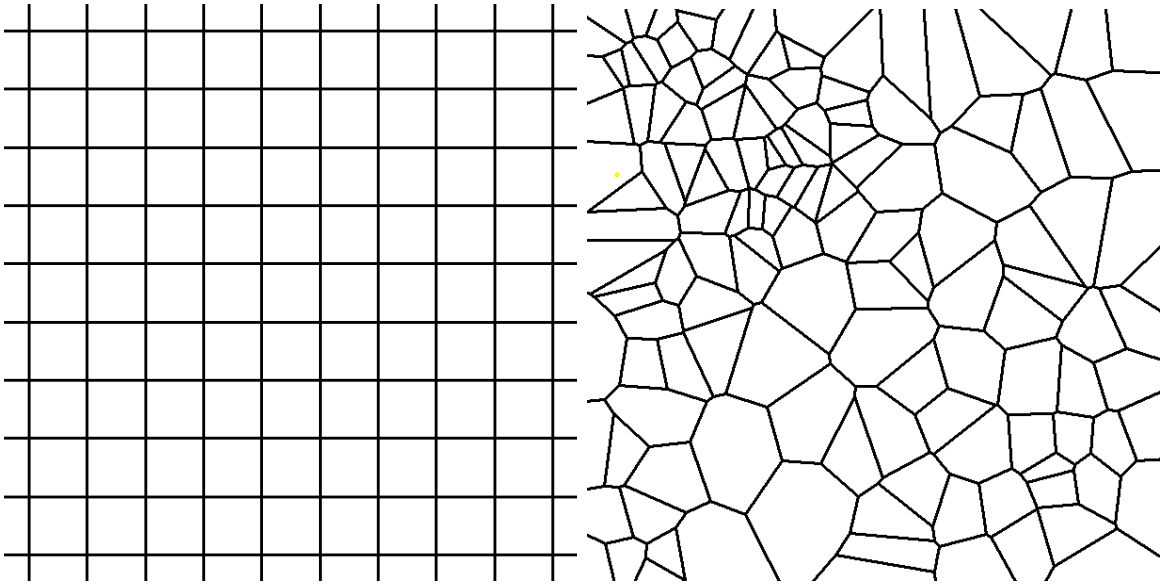
These simulators cannot be modified to access mobility data due to licensing. A secondary issue is that applications and communications protocols cannot be layered above such traffic simulators. PARAMICS, CORSIM and TRANSIM have been noted to use the TIGER road database [TIG04] which provides maps below an accuracy of 50 meters. However, road data is provided without meta-data (e.g. speed limits, traffic direction, road altitude, traffic lights).

Random Waypoint models typically choose a random destination for a specific vehicle and then route the vehicle across a road network [Bet01b, PGHC99]. The usage of Random Waypoint models, while widely used and incorporated in some simulators like the Scalable Wireless Ad-hoc Network Simulator (STRAWS) [CB05], was considered harmful to VANET studies by Yoon et al. [YLN03]. More modern synthetic models have used 'macro-simulators' to construct traces using city maps and GPS coordinates - vehicles being constrained by the irregular road networks of differing cities. Mobility simulators, have in some cases attempted to simulate traffic within city road networks. The traffic flow simulator Videlio was used by Lochert et al. [LHT+03] to produce a number of traces for the city of Berlin (6.25 km x 3.45 km area).

The micro-mobility simulator CanuMobiSim[2] is a standalone mobility generator [THB+02, SMR05]. It includes a collection of physical and vehicular mobility models, and has been used to generate synthetic traces. Given the availability of real road topologies provided with geographical data files, the simulator generates traffic by creating and assigning a number of routes using Dijkstra's shortest path algorithm [Dij59]. An alternative mobility method creates attraction locations which vehicles attempt to gravitate towards. Traces generated can

[2]http://canu.informatik.uni-stuttgart.de/mobisim

be used in QUALNET and NS-2 (Section 2.5.2). VanetMobiSim[3] by EURECOM provides an extension for macroscopic and microscopic vehicular mobility [FHFB07]. VanetMobiSim is capable of generating Voronoi tessellation maps to mimic city road structures (Figure 2.6(b)). Road detail includes separate directional vehicle flows and the identification of intersections and overpasses. The combination of CanuMobisim and VanetMobiSim provides a highly 'realistic' set of models.



(a) 10 x 10 Grid. (b) Voronoi.

Figure 2.6.: Generated road topologies.

Naumov et al. [NBG06a], using the multi-agent microscopic traffic simulator (MMTS), extended synthesis to include a number of driver behaviours to trace generation over 24 hour periods. Driver behaviour was influenced by perceived road congestion. 260,000 vehicle traces were generated for the city of Zurich for a 24 hour period. Since this work a number of traffic and mobility simulators have become available including SUMO [KSB09], Traffic and Network Simulation Environment (TRANS) [PRL+08], Mobility Model Generator for Vehicular Networks (MOVE) [HFB09], VanetMobiSim [HFBF06] and others [HFB09]. Tonguz et al. [TVB09] propose a cellular automata approach to construct traffic mobility model. A benefit of trace synthesis is that one can apply many different patterns with some random seedings; however traces may not be real.

[3] http://vanet.eurecom.fr

2.5.2. VANET simulation

Some widely used VANET simulators include Network Simulator 2 (NS-2) [GC08, MI04], QUALNET [ABFeH07], GrooveNet [MWS+05], OPNet [Cha99], Opportunistic Networking Environment (ONE) simulator [KOK09], CSIM [Sch88] (Lockheed Martin), as well as others. Argonne was an early ITS simulator for centralised systems [EDH+96]. We detail a subset of simulators which are relevant to V2V service simulation.

Argonne Simulation Framework: One of the oldest parallel microscopic simulation frameworks is the Argonne Simulation Framework (ASF) developed by Ewing et al. [EDH+96] which was specifically concerned with ITS applications. The ASF aimed to provide a platform for large-scale and centralised ITS services. These services used a centralised controller for vehicle management. Within the framework vehicles with built in navigational computers were connected to a Traffic Management Center (TMC) via data links. Road and weather data was periodically updated and associated with road map data, providing dynamic route planning. The TMC would track vehicle movements and provide advisories to drivers using a detailed graphical display to achieve traffic control. Argonne also attempted to 'realistically' model human response times by considering vehicle details, road conditions, weather and driver personality type. A detraction of the ASF was that it is highly service specific.

Network Simulator 2 originally developed at University of California-Berkeley, is an object orientated discrete event simulator. The simulator and its components are written in C++. A secondary language, OTcl is used to script simulation scenarios, specify scenario events and protocols. Simulation results can be either analysed further or visualised using various post-processing tools. NS-2 considers small scale simulations. To simulate ad-hoc networking scenarios, NS-2 relies on a collection of extensions to incorporate mobility models. For example the National Highway Traffic Safety Application (NHTSA) was constructed as an extension to consider DSRC standards. While NS-2 has been widely used to model routing protocols for both static and mobile networks, a detraction from using NS-2 is its complexity. Application scripts must be written using OTcl. OTcl is not directly portable. The addition of new components or the modification of existing components requires training in both C++ and OCL.

QUALNET[4] is a feature rich proprietary network simulator [ABFeH07]. QUALNET as an extension, enables simulations to consider Mobile Ad-hoc Networks. The simulator provides performance measurements for scenario based ad-hoc networks using an OSI layered architecture. Each layer in the model is itself an object with its own variables and structures. Messages transferred between nodes (vehicles) are exchanged between layers. QUALNET is modular, thus enabling simulations to be run on parallel distributed systems. The simulator supports IEEE 802.11a/b/g and WiMAX layers, however, a disadvantage of QUALNET is such that it is non-service specific. Mobility models are not easily incorporated into the simulator. Similarly, the proprietary nature of the software makes it difficult to amend specific aspects of the simulator.

Optimized Network Engineering Tools (OPNET)[5] is a proprietary discrete event simulator for the simulation of system processes [Cha99]. Its initial use was the simulation of static networks. OPNET provides a number of useful tools for the simulation of ad-hoc systems. Recently, the tools for mobility specification and integration have improved. OPNET simulates low level networking and physical layer operation within the OSI model. A disadvantage of OPNET is one of processing requirements. Simulations, due to their highly detailed simulation of the radio can consume more time than otherwise is common with other approaches. To counter this, OPNET can be executed in parallel, on clusters of machines.

OMNeT++ [6] is a discrete event simulation environment which can be used to build V2V simulations. OMNeT++ provides a component architecture for models. Components are programmed in C++ and used to build models using a high-level language known as NED. To improve its ease of use, OMNeT++ provides a number of graphical user interfaces. While a number of extensions offer mobile networking options, the 'Mobility Framework'[7] is commonly used for the simulation of VANET routing protocols. As more extensions are added, OMNeT++ will become increasingly popular as a network simulation platform as it provides a significant number of tools for simulation analysis and visualisation.

GrooveNet [8] is a self contained simulation tool which models V2V communications, incorpo-

[6] `http://www.omnetpp.org`
[7] http://mobility-fw.sf.net
[8] `http://www.seas.upenn.edu/~rahulm/Research/GrooveNet`

rating mobility models constructed from TIGER road maps and topographies [MWS+05]. At initialisation, each vehicle is associated with its own mobility model. The simulator is capable of simulating thousands of concurrent vehicles and provides a method of hybridising experiments to allow one to attach real vehicles to the simulator while simulating further vehicles. GrooveNet uses a geographic map model to position vehicles, providing a collection of four mobility models and a selection of communication models for simulation. Furthermore, the mobility model allows one to stipulate the source and destination addresses of a particular vehicle (as a selection of waypoints). Communication models are capable of injecting faults in message retrieval, such as dropped packets. GrooveNet however lacks a method of outputting traces which may be used in comparable simulators.

2.6. Vehicular Frameworks

Vehicular frameworks provide scaffolding for the construction of vehicular applications. We highlight two specific frameworks by Kim [Kim07] and Leontiadis [LM07, Leo09, LCM09a, LCM09b] as they focus on traffic data. The framework by Kim [Kim07] is used for traffic information dissemination. Kim's motivation is that other simulation approaches to VANETs do not provide frameworks which support both transportation issues and communications within a single framework design. The framework is specifically structured to consider traffic data dissemination performances in VANETs. The framework considers low level data dissemination and routing for vehicles in grid scenarios. Kim uses a combination of two propriety systems, PARAMICS and QUALNET to simulate the framework. Within the approach, PARAMICS provides microsimulation mobility. These mobilities are connected directly to the QUALNET network simulator, allowing the framework to simulate IEEE 802.11a messaging. Mobilities consider variable speed vehicles, allowing vehicles to over-take and re-route based on traffic levels perceived within the road network.

Work by Leontiadis et al. [LM07, Leo09, LCM09a, LCM09b] provides a content dissemination framework for vehicular networks which uses *push* and *pull* techniques. In comparison to Kim, Leontiadis focuses on the control aspects of push and pull dissemination strategies in vehicular environments. Leontiadis considers centralised traffic authorities, who use a mixture of vehicles and 'info stations' to collect and share data about a city. The framework is used to route and

maintain information in specific geographic locations. Pushing data broadcasts content to a subset of other vehicles. Geographic data is used to aid dissemination and filter messages, with vehicle mobility patterns being predicted to improve dissemination. Notably, the framework was tested on a small set of vehicles driving on a real road network.

2.7. Intelligent Transportation Systems

From a global view, Intelligent Transportation Systems (ITS) represent 'technologies and systems engineering concepts to develop and improve transportation systems of all kinds'[9]. By definition, ITS encompass a broad set of computing systems and technologies which seek to enhance road networks. ITS approaches are often non-specific and could be applied to road networks, shipping lanes and aircraft. ITS include both passive management systems (e.g. toll collection, emergency alert beacons, traffic signals, navigation, passenger information, speed cameras and monitoring systems) and active data systems (e.g. payment systems, weather information systems, travel and assistance systems, adaptive cruise control, fleet management).

Many implemented solutions are not specifically highlighted as 'ITS' solutions as ITS lack both a coherent taxonomy of paradigms and as such are not specific in the development of services, and many methodologies are highly fragmented. One of a set of examples of a real world centralised traffic control systems is that provided by the Tokyo Traffic Control Center (TTCC) [Ino76]. TTCC computers are fed sensory data from a number of sources, including cameras, vehicle counters and inductive loops embedded within the road network. 15,154 traffic signals and over 1000 intersections are sensed and controlled by human operators using various strategies to reduce congestion along city roads. As well as directly controlling traffic lights and signals, the controllers update drivers with traffic information using FM radio stations and traffic information boards. Similar approaches to the TTCC have been undertaken by other government agencies including the Houston TranStar (HTS) in the United States [Sus05] and the Highways Agency (HA) [Vic00] in the United Kingdom. The HA while similar to the TTCC also provides drivers with traffic forecasts using data analysis. Such approaches are large and highly dependent on a large infrastructure of sampling sensors.

There are large costs associated with their setup, operation and maintenance. Sensors are

[9] http://ewh.ieee.org/tc/its

positioned at strategic locations along the road network. The size of a city makes the roll-out of sensors over the entire road network prohibitively expensive. Hence such agencies often confine their traffic control to highways. However all roads are interconnected and each adjacent road section can influence its neighbouring section. Furthermore applying feedback (influencing drivers to change their route or drive) is challenging since not all vehicles using the road system are likely to be aware of changes in traffic or congestion. This limits control centres to using information boards and traffic lights as a means by which to manage traffic. Such systems represent an extreme set of centralised solutions. Given the large costs and complications associated with monitoring and controlling traffic various 'hybrid' solutions have been proposed and some have been implemented.

A significant set of work has sought to collect data about the road network using mobile phones and vehicles. We specifically focus on vehicle data collection. CarTel [HBZ+06] collects, processes, delivers, and visualize data from sensors located on mobile units in vehicles. A CarTel node gathers and processes sensor readings locally before delivering them to a central portal, where the data is stored in a database for further analysis and visualization. Nericell (Traffic Sense)[MPR08] and Pothole Patrol (PP) [EGH+08] provide 'rich' sensing data, detecting and reporting various road conditions including traffic. In the case of Nericell, data is provided using mobile phone accelerometers, microphones, cellular radios and GPS positions. Phones detect potholes, bumps, braking, and 'honking'. Pothole Patrol vehicles are specially equipped with a multitude of sensors which gather data about roads. Potholes are identified using simple machine-learning techniques. Various online communities have been formed to share location and map data (crowd source), these include Open Street Map (OSM) [Goo07, Ope11] and works from the MIT Senseable City Lab [CRK08, PHL+10, CCL+11]. The OSM community is a volunteer collaboration effort for the generation of detailed worldwide maps. Senseable City uses crowds of users to collect and share data via a centralised authority (or central systems). WikiCity [CRK08] and similar approaches [CCL+11] advocate the use of a community of citizens to collect and share data about the city using mobile phones. The MIT Syn(c)ity [MKDL+11] (and Affective Intelligent Driving Agent) proposals seek to provide a centralised computer guide to a driver, recommending routing throughout a city based on driver statistics, preferences, their social network, the state of the vehicle and known traffic conditions.

Traffic control and feedback approaches include the Mobile Millennium Project [HWH+10],

the Waze crowd-sourcing service [Waz11], Google Latitude [Bar09]. The Mobile Millennium Project (MMP), undertaken by UC Berkeley in San Francisco, implemented a traffic-monitoring system using more than 5000 mobile phones. The mobile phones were associated with individual vehicles. Sets of vehicles gathered GPS position data and state data about the road network. Data was uploaded to a central authority via the mobile phone network and processed. The central authority's analysis of the road network was sent back to vehicles in real-time. Data was collected by managing 'virtual trip lines' or data collection points to improve the privacy of users. In effect the system analysed short fragmented GPS traces. Similar approaches have been undertaken by both Waze [Waz11] and Google Maps [Bar09], where user trace data is used to approximate the traffic occurring along a road network. Waze differs in that it includes community broadcasts from drivers (for example, users may report seeing a traffic accident at a location).

Such approaches use vehicles and drivers to sense the state of the road network over time. MMP, Waze, GL, WikiCity and others are feedback approaches, requiring that a central authority (server) continue to exist to provision traffic control. Criticisms levelled at centralised systems include issues failure, maintenance and privacy. However, if the central authority is unavailable then the service is unavailable. The authority is required to maintain a central system to service users of the service. The centralised nature of the system requires that a service provider collect and analyse data. Users need to be sufficiently trustworthy of the provider as their location data is often considered sensitive and private to many users. Furthermore, users need to able to trust the guidance being provided by the authority itself. As with many distributed approaches, the service requires that data be crowd-sourced. As more users use the service, we expect more data sources and hence improved analysis.

In contrast to centralised approaches, decentralised solutions have constructed small scale vehicular services on VANETs. The Mobile Environmental Sensing System Across a Grid Environment (MESSAGE) initiative by Imperial College London has sought to sense pollution within cities [HLP11]. Vehicles use standard WiFi (IEEE 802.11g) and WiMax (IEEE 802.16) communications to share and upload data to a central server when the opportunity arises to do so. Effectively the VANETs produced allow vehicles to collect data using delay tolerant routing approaches. Similar small scale systems have been produced by the Cooperative Vehicle Infrastructure System (CVIS) [GPR+10] and CARLink [SN09] and the CAR2CAR consortium

Approach	*Architecture*	*Sensors*	*Comms.*	*Advice*
TTCC [Ino76] HA [Vic00] HTS[Sus05]	centralised	static	POTS	indirect
CarTel [HBZ+06] PP [EGH+08]	centralised	mobile	GPRS	none
Nericell [MPR08]	centralised	mobile	GPRS	none
OSM [Goo07, Ope11]	centralised	mobile	3G	none
Senseable City Lab [CRK08, CCL+11]	centralised	mobile	3G	indirect
MMP [HWH+10] Waze [Waz11]	centralised	mobile	3G	direct
GL [Bar09]	centralised	static & mobile	3G	direct
MESSAGE [HLP11] CVIS [Kom10]	decentralised	static & mobile	WiFi & WiMax	none
CARLINK [SN09]	mixed	static & mobile	WiFi & WiMax	indirect

Table 2.1.: A comparison of implemented systems.

[LMP+07].

CARLINK[10] was developed to provide a wireless traffic service platform between the cars [SN09]. Both V2V and V2I interactions were considered. As with MESSAGE, static road-side base stations were used to collect vehicle data such as weather and traffic data. This data was stored in a central database. The database was then used to push updates back to the vehicle passing the base station. Two significant applications were developed: (a) the FSF (Finding and Sharing Files) service and (b) a puzzle bubble game. The applications were constructed in a Java based application called JANE that provided a means of testing the usage of real vehicular services using the CARLINK model. CARLINK has successfully constructed small and working V2V networks which comprise a hybrid of V2V and V2I interactions. Large-scale vehicular systems have not been constructed.

Table 2.1 identifies a number of characteristics of the systems in terms of centralisation versus decentralisation, the provision of mobile and static sensors, requirements for a data network infrastructure and whether the service can directly (e.g. messaging individual vehicles) or indirectly (e.g. information boards) advise vehicles about changes. By classifying these systems it is apparent that no single implementation considers operating a decentralised advisory application using V2V interactions.

[10] http://carlink.lcc.uma.es

2.8. Summary

In this chapter we considered related work which has influenced and motivated our vehicular framework. We began the chapter by highlighting the cost of inefficiency within the road network and the need for low cost traffic management in cities. Emerging technologies are making ITS and similar services cheaper as time progresses; however, even given the reduced hardware costs, the relative costs of building centralised traffic control services are prohibitively high for many cities. We highlighted some real world examples and discussed the use of vehicles as sensors within the city to reduce the costs associated with such systems. Major investments need to be made in sensory infrastructures as well as communications networks for centralised authorities to operate services. Systems must be set up, deployed and maintained. Many cities are large covering significant geographic areas.

With the advent of V2X technologies it may be possible to remove the necessity for a centralised authority and build more *decentralised* services. These services are *broadcast based*, *decentralised* and *scalable* and use V2X interactions to construct and supply a vehicular service. In the remainder of the thesis, we present and motivate a new framework for the development and evaluation of decentralised vehicular services, which uses scaled vehicular interactions and mobility to provide vehicle-based applications and services.

3. Decentralised Vehicular Services Framework

In this chapter we present the decentralised vehicular framework which features a geodetic (geodesy) based scaffolding methodology for developing decentralised vehicular services. Spatial and time based calculations are used to record and estimate fine grained mobility patterns. Decentralised vehicular services reactively leverage shared mobility fragments, messaging and feedback loops to repeatedly adapt state data stored about the world, thereby providing traffic management services. The framework makes it easier for developers to develop vehicular services such as decentralised traffic management services. The inclusion of simulation within the framework provides a means of improving our confidence in a prototyped service. Furthermore, the combination of framework specific features (particularly dynamic mobility in combination with messaging and feedback loops) are not available in other vehicular frameworks.

The chapter begins by defining the framework's requirements (Section 3.1). We then describe the framework model, as well as the layered world sub-models and the vehicle model (Section 3.2). The section also describes the specification of mobility in terms of *routes*, *plans* and *fragments*. Finally, we discuss the simulation methodology used to evaluate prototyped decentralised services constructed using the framework (Section 3.3). The framework is used as a tool to develop and evaluate mobility-based decentralised traffic management services in subsequent chapters.

3.1. Requirements

Our framework is concerned with addressing the following requirements:

Mobility. Previous frameworks have considered mobility as *permanent* and related works

have largely focused on the message routing overlays produced by vehicles over time (for example message routing protocols [ABFeH07, BMJ+98, BGJL06, LW07]). In these scenarios, vehicle mobility is predetermined. In other words, a vehicle is expected to adhere strictly to the route chosen at the start of travel. Vehicles do not react to data which may be available within the environment. Decentralised vehicular services adapt to changes in their environment and this includes mobility adaptation as services share mobility data with one another. Hence we require a mobility specification which is *non-permanent*, *flexible*, *easily understood* and *fine-grained*. In this context, fine granularity (or high resolution) is necessary as vehicular services need to consider precise geographic distances as well as precise vehicle movements over time. Error tolerances are also limited. For example, vehicles pass each other on dual carriage-ways with very little distance between them. Flexibility is required as vehicles may seek to change their mobility at run-time; communicating these changes is necessary given limited messaging capabilities. A trade-off of mobility flexibility is complexity. A variety of changes may occur and vehicles may change their mobility rapidly within small measurable distances. We derive a number of complex mobility functions from raw geographic position and time data. We address fine-grained and planned mobility in Section 3.2.3.

Decentralised Control. Vehicles are required to serve and provide services to themselves and one another in a series of ad-hoc contacts. Vehicles act as sensory inputs and users of data. Each vehicle acts as a component of a larger vehicular community. Vehicles are required to assess inputs and effect the desired output actions specified as a group behaviour. To achieve this vehicles are required to handle actuation and messaging events between themselves, neighbouring vehicles and their environment. We require a means of specifying control and a set of interfaces by which we may interpret and effect control. A protocol provides a procedure for regulating messaging and control between vehicles. Vehicular services are the result of sequences of inter-vehicle contacts and multiple decision making stages within the message passing process. We assess how vehicles specify service control in Section 3.2.3.

Cooperation and Scalability. To operate, decentralised services require scaled numbers of interactions between populations of vehicles. The ability of a decentralised service to

support a large numbers of vehicles without impeding service performance is referred to as *scalability*. Vehicles are required to *cooperate* with one another in the provision and usage of services. In other words, vehicles work together to provide a correctly operating service. Without cooperation decentralised services are limited in their ability to affect change or achieve goals and thereby become non-beneficial. As increasing numbers of vehicles use the road network and wireless radio communication, a number of challenges occur in terms of resource usage and optimisation.

Performance. The performance of a decentralised vehicular service should seek to better the performance of a centralised or competing approach with similar or reduced performance requirements, costs and overheads. Measurements used to compare performance between competing centralised and decentralised services are often problem specific; however, a number of performance measures are more global in their applicability, notably those relating to messaging and mobility. The framework requires a means of measuring and comparing performance between competing approaches. As *metrics* are specific, we compare performance individually in the context of two traffic traffic scenarios (Chapters 4 and 5).

Fault Tolerance. The road environment in which vehicles operate is highly dynamic. As such a decentralised service must adapt to faults which might occur during run-time. Inducing faults in various components of the framework is necessary to consider how a service adapts to faults, for example messaging faults and unexpected mobility changes. Decentralised services are required to operate in the presence of fault, resolving fault on-the-fly. For example, the exit of a vehicle from a community of vehicles should not incur a significant loss in service performance, and vehicles should be capable of recovering from such a fault. Similarly, the ability of a service to tolerate change during operation is scenario specific. We consider tolerances in more detail in Chapters 4 and 5.

Extensibility. The framework is required to include mechanisms by which we may enhance and improve the modelling of vehicular services, thereby improving our confidence in a service operating correctly at deployment. As such, the framework is coupled with simulation for modelling. Due to resource constraints, the model chooses core features deemed most important for the modelling of vehicular services. For example the mod-

elling of vehicles and the messaging between vehicles. The more complex the model, the more computationally intensive the framework. The model used provides a confidence in understanding the feasibility and operation of a service. To improve confidence the framework needs to be extensible. The simulation approach used is modular and we discuss its use in Section 3.3.

Some orthogonal qualitative requirements are not addressed within the framework. They include:

Trust. The definition of trust within computing is varied, yet a core requirement of systems and services. The framework assumes that all vehicles within the model are 'trustworthy'. We assume that vehicles acting within a service 'honestly' collect, share and use data. We assume that a trust model might be overlaid on a decentralised service. For example decentralised social trust systems are one such approach [Gol10]. A lack of trust can have significant effect on cooperating systems, forcing vehicles to limit their interaction. The accuracy and integrity of data is called into question and the quality of service provision as well. Trust is itself a large problem and beyond the scope of the framework presented.

Privacy. Within decentralised vehicular services, data is shared between vehicles for the creation of a service. As data is broadcast wirelessly, it is possible for eavesdropping to occur. Data and the detection of transmissions and interpretation of mobility data may be used to infer personal information over time about the details of a particular vehicle or a subset of service users. As with trust, the privacy issues relating to vehicular services are not assessed with the thesis.

Incentives. The incentives or rewards for decentralised vehicular services are not specifically detailed, however we assume that vehicles may be motivated to use vehicular services for their own benefit (not necessarily the benefit of the entire community i.e. altruism). The problem of incentive is itself a larger problem. There are clearly questions to why a driver would be inclined to become part of a community of vehicles if becoming part of that community does not benefit them. Vehicles are required to expend their own resources (for example communications, privacy, energy and storage) to enable service provision. Decentralised services do attempt to reduce the costs of traffic management,

but for those costs to be shared amongst users the users need to adopt the technologies, either by choice or by requirement.

3.2. Framework

In the previous section we considered the requirements of a decentralised vehicular framework. The decentralised services presented are expected to exhibit complex behaviours which are the consequence of feedback loops, cycles, shared mobility data, varying inputs and outputs, and adaptation. Our requirements seek to focus the framework on providing abstractions for mobility and control, cooperation and scalability as well as the accurate measurement of the performance of a service in relation to both decentralised and centralised approaches. In the following sections we detail the decentralised framework.

3.2.1. World Model

Vehicular services are required to operate in a real and physical setting. Simulating such a perfectly real setting is computationally infeasible within simulation due to complexity. Hence, we extract a subset of real world features and model these features attributes, characteristics and abilities as a set of layered sub-models which in combination construct a world model. In contrast to static networks, vehicular networks are mobile and ever-changing. This adds extra complexity to models. For example, we repeatedly re-evaluate the position of vehicles spatially to determine whether those vehicles are within proximity of one another.

The approach used by the framework is modular in its design to achieve extensibility. Inhabiting the world are a collection of mobile and static objects. Vehicles represent a subset of mobile objects. In terms of the world model, vehicles represent processing and communicating elements. Each vehicle executes its own protocol(s) in timed execution cycles (however, protocols are not required to execute at each cycle). In effect the framework model comprises a bounded but discrete geodetic *mobility and messaging model* and a separate feedback driven *vehicular architecture* (Figures 3.1 and 3.2). World position within the framework uses the World Geodetic System 1984 (WGS84) standard [NDM+89, Tru04]. The world model consists of three interacting layers: a *transportation layer*, a *mobility layer* and a *messaging layer* (Figure 3.1). Included in the model is a discrete *clock* to mark transitional time-steps. As the

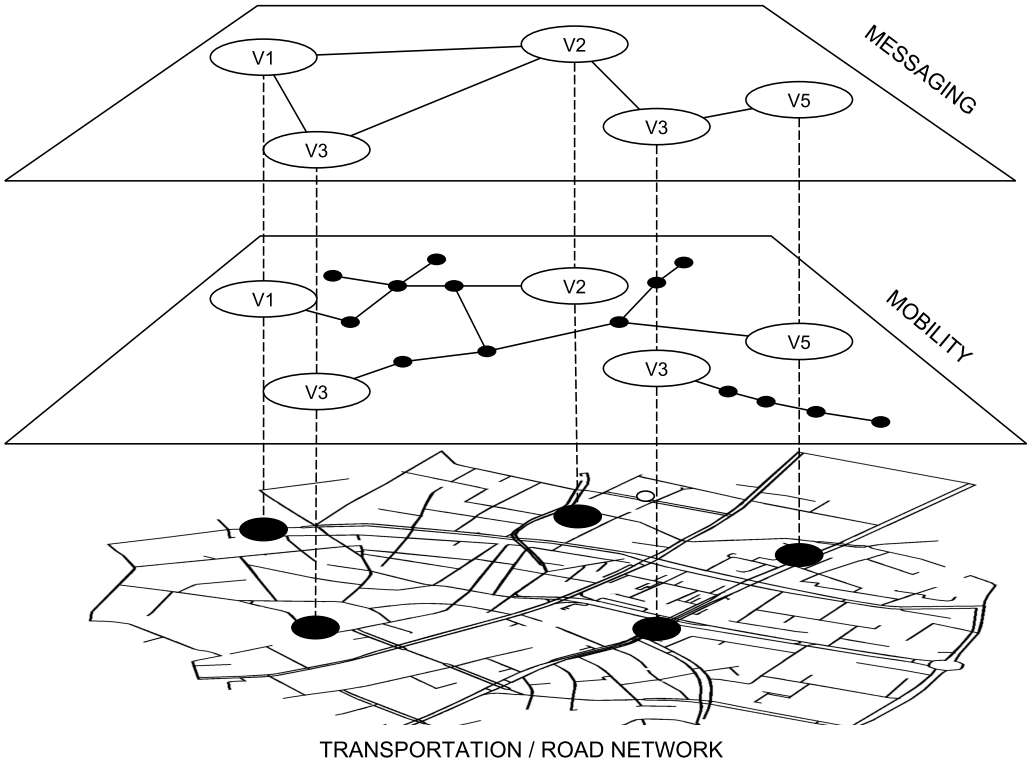


Figure 3.1.: Layered model: vehicles (V1 to V5) exist on a transportation network. Each vehicle records its past mobility as well as routing information on how to reach its destination using the road network.

approach is geodetic, all objects residing in the world use a common waypoint (W) specification to define geographic position on the surface of the Earth:

```
waypoint = (latitude, longitude, altitude)
```

For example:

```
waypoint = (51.498538,-0.176468, 123m)
```

This usage of waypoints allows us to consider position as it would be considered in the real world space (inside a city road network). In other words, a virtual mapping of a vehicle in simulation corresponds with an actual world mapping. Algorithms which calculate precise geographic position typically require more computing resources and time.

However, there are some advantages to using geographic positioning. As vehicles and services use maps, geographic data and GPS standards to define various layers, we need not convert or lose granularity when using geodetic sources. For instance we can overlay road maps atop

real world geographic topologies, which more accurately specify the real world road networks vehicles are travelling. Services are directly portable. Stipulating spatial positions is more realistically considered in terms of the real data available to a vehicle in time. Vehicles themselves use the world model as a reference by which they may note their location as waypoints, sense attributes about the space and message one another by querying various layers. Higher layers are dependent on the lower layers.

Transportation layer. This layer represents the base dependency layer. Data which defines the transportation layer is provided by road maps. Vehicles mobility is constrained by the road network. However, the framework relationship between mobility and the road network is loose such that road networks act as guides for vehicle mobility. The transportation layer models roads in terms of their meta-data such as point-to-point direction and speed limit. Other timed constraints can also exist for intervals of time. For instance, a road may only be available for use during particular hours of the day. By referencing the transportation model, mobility is specified within these contextual constraints.

Mobility Layer. This layer represents the movement of a vehicle as a set of *past*, *present* and *future* positions in the form of traces. The split of mobility and transportation layers mean that mobility is *flexible* and dynamic (changeable at run-time). Each vehicle monitors and amends its own mobility state for every time-step (t). We discuss mobility as it is related to the vehicle in more detail in Section 3.2.3.

Messaging Layer. This layer models wireless radio communication between vehicles. Again, constraints are used to model the broadcasting and receiving of messages according to the known particulars of a given wireless standard. Manipulating the layer has the ability to change the characteristics of messaging. For example, inducing dropped messages.

The combination of these layers seeks to represent the mobility and messaging events experienced by an individual object existing in geographic space over successive time-steps. Notably, the mobility of an object is dependent on the constraints of the transportation network. In turn, the derived messaging network is dependent on the mobility of vehicles and the constraints applied at each layer. As the world model is modular it allows us to classify and apply

different controls at successive layers. For example, changing the maximum communication range of a message would be applied by modifying the messaging layer or reducing the speed limit might be modified by adjusting the transportation layer.

3.2.2. Time

In reality, time is continuous. Within the framework, time is considered discrete for the purposes of simulation. The framework assumes that a global synchronised clock exists at a present time-step (t). Furthermore, we assume that vehicles and objects are synchronised with this clock. Erroneous clocking has the potential to negatively affect a service, however the effect is dependent on the clock interval and the service. We use an offset (k) to specify past and future times. The past is denoted by negative offset time ($t-k$), while the future is denoted by positive offset time ($t+k$). The interactions between vehicles are not modelled continuously; rather signals such as messages need time to be received in a next time-step. It holds that a signal sent in the previous time-step ($t-1$) is received only in the present time-step (t). Clocking modelled the IEEE 802.11p standard Wave Short Message (WSM) broadcast rate.

3.2.3. Vehicular Model

In the previous section we considered the world model in which both static road-side objects and vehicles exist. In the following section we consider the vehicular model. Figure 3.2 shows the component architecture used by both vehicles and static objects.

Objects and Vehicles

Objects and vehicles represent the atomic constituents of the framework model. We call the collaboration of vehicles a *service* (i.e. it requires two or more vehicles to enable or enact a vehicular service). Specifically, these elements provide a means of architecting decentralised services by separating the world model from protocol design and execution. Base objects are deemed static (non-moving) elements, while vehicles are considered derived mobile elements (movable). Both are capable of executing a protocol and thereby a service. Protocols retain core reactive functionality for dealing with messaging and sensory inputs. Figure 3.2 shows the object and vehicle architecture which consists of a selection of components. The operation of the architecture is cyclic. Each vehicle protocol execution loops periodically.

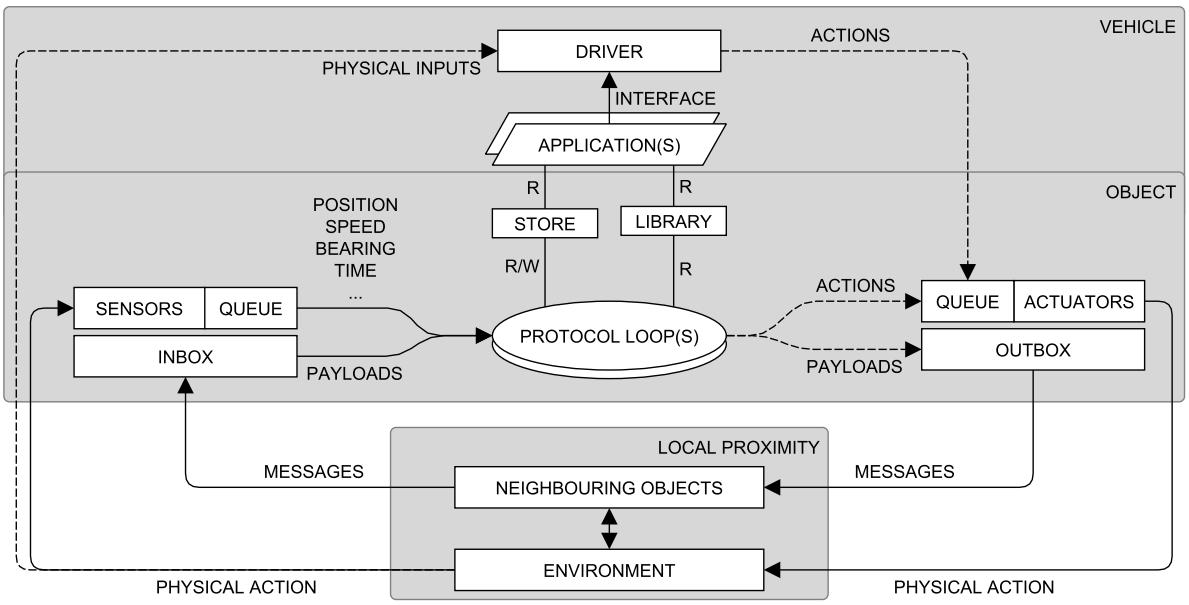


Figure 3.2.: Vehicle architecture: a specialised extension of an Object. The architecture shown does not include management components required to manage protocols and applications.

In each time-step, received messages and sensory data from the local proximity are sampled. Sensory inputs are queued in the sensory queue, while retrieved messages are queued in a message *inbox*. A polling call provides the messages and sensory data to the *protocol loop* for processing as parameters. For each cycle multiple protocols may exist simultaneously. Message payloads are matched to protocols. The protocol loop is also provided a interface to access *library* functions and a read-write *store* to store successive data beyond the present execution cycle or time-step (t). An *application*, interfacing with the protocol loop provides a *driver* to a vehicular service based on the data which has been collected, processed and shared between vehicles. We assume that actions may be taken either automatically by the vehicle or by the driver.

At the end of each cycle, queued actions are actuated and queued messages are broadcast. If other vehicles are within local proximity of the actions and message broadcasts of a vehicle, then they may act on these new data. As such, two feedback loops are created between broadcasting and/or actuating vehicles and listening and/or sensing vehicles. Where listening/sensing vehicles do not react the feedback loop is broken. Notably, vehicles monitor and manage messages to ensure that self-cyclic messaging does not occur (self messaging from previous time-steps via neighbours).

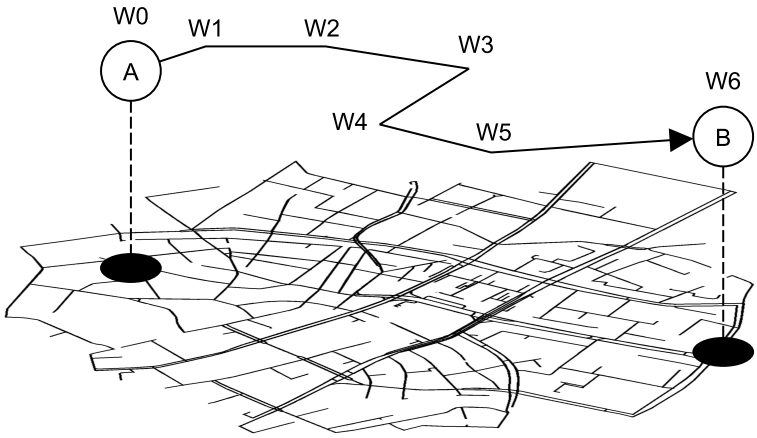


Figure 3.3.: An example route for a single vehicle travelling between A and B signified by the sequence of waypoints ($W0$ to $W6$).

Mobility and Fragments

Within our framework *mobility* is defined as the change in position of an object over a series of time-steps. Mobility is assumed to be dynamic, such that it may change as a vehicle travels the road network. We assume that mobility is unique to every vehicle, specifically as vehicles are not allowed to collide with one another over time, but also because different vehicles typically attempt to travel from source (A) to destination (B). Mobility data is held within the *store* (Figure 3.2).

A method of mobility specification is a requirement of the framework as vehicles require mobility data to travel and vehicular services share mobility data to provide decentralised services to themselves and one another. The mobility of a typical vehicle (V) is constrained by the road network. The attributes of position, proximity, bearing (orientation) and speed can all be sensed or derived by an object or vehicle which may also access positioning and time data, for example GPS data. We assume vehicles journey from a source to a destination. Such a journey is specified as a route R, which is a totally ordered list of n waypoint positions where $n \in \mathbb{N}$. For example we may write a R as the sequence:

$$R = \Big[W0, W1, W2, W3, ..., Wn\Big]$$

Figure 3.3 depicts one such possible road network and R consisting of a sequence of waypoints connecting two geographic waypoints A and B. R is typically coarse in its description

of movement. When associated with time data, a route (R) is considered a *plan* of how a vehicle should travel and when it expects to travel a particular waypoint. An infinite number of intermediary positions are possible between waypoint elements inside the route. A processed form of R, called a *track*, associates a sequence of positions with a unique clock time (t). Waypoint-time pairs of the form (waypoint,t) give the expected position of a vehicle in time, where $t \in \mathbb{Z}$ given that the current time is t. The example T_{c+f} combines current (c) and future (f) waypoint-time pairs. Time-steps follow that $t \leq u \leq v \leq w \leq z$. These time-steps are associated with a set of unique waypoint positions found in R. The association implies that a vehicle *was*, *is* or *shall* be at a specific position in time.

$$T_{c+f} = \begin{bmatrix} (W0, t) \\ (W1, u) \\ (W2, v) \\ (W3, w) \\ \cdots \\ (Wn, z) \end{bmatrix}$$

As such, mobility is specified for each vehicle as an in-order sequence beginning at a start position and ending at a destination. Past mobility is recorded as a *trace history*. The coarseness of a simple track creates a granularity problem as we need to specify precisely the continuous positions of vehicles in intermediate positions and times. Let us suppose that time progresses to a new current time-step (t). Future mobility is estimated as a *future track* (f). Supposing that a vehicle travels an ordered track of unique positions from $W0$ to $W6$ beginning at a previous time, a trace history for a routing from the present position $W3$ through to the previous position $W0$ in n-many previous time-steps from the present time may be represented by the trace history h. Similarly a future track for a routing from $W3$ to $W6$ in n-many future time-steps may be estimated as f. Increased fragments typically add granularity to mobility. The comparison between h and f is illustrated as the ordered set of array values:

$$h = \begin{bmatrix} (W3, t) \\ (W2, t-k) \\ ... \\ ... \\ (W0, t-n) \end{bmatrix} \quad f = \begin{bmatrix} (W3, t) \\ (W4, t+k) \\ ... \\ ... \\ (W6, t+m) \end{bmatrix}$$

The complete mobility of a vehicle is hence a union of past, present and future positions ($h \cup f$). A recognisably different characteristic of f is that of its *generation* and *estimation*. To generate f requires the usage of an algorithm to estimate the position of a vehicle in future time-steps. The accuracy and quality of f is hence dependent on the algorithm used to estimate future movement. There may be multiple future tracks to a destination and estimation requires processing. Regardless, a requirement of both future track and trace history mobilities are that they remain associated to a specific position on the road network at a specific time. Subsets of the tracks are considered mobility *fragments*. Fragments represent pairs of waypoint-time tuples. For example the set $[(W0, x), (W1, y)]$ is such a mobility fragment for two arbitrary times x and y, where $x \leq y$. We may infer a great deal of information from fragments. The case studied vehicular services considered in later chapters use mobility fragments as a means by which to provide services. As these mobility fragments represent intervals between waypoints they represent both past and predicted future mobility.

Sensing and Actuation

A variety of sensors and actuators could be provided to a vehicle protocol; however the framework currently models just two sensory inputs, namely positioning data (*waypoints*) and clock data (`t`). Each protocol polls sensory inputs from the sensory queue and the local clock every time-step. We derive complex mobility data from these base inputs. For instance, for every time-step beyond the first time-step, a vehicle can calculate a `bearing` and `speed` by referring to h and sundry stored mobility fragments. Bearing refers to the navigational orientation of a vehicle (also known as the forward axis), measured in degrees. By default the framework considers bearing in terms of relative bearing (the direction of the vehicle as it is travelling).

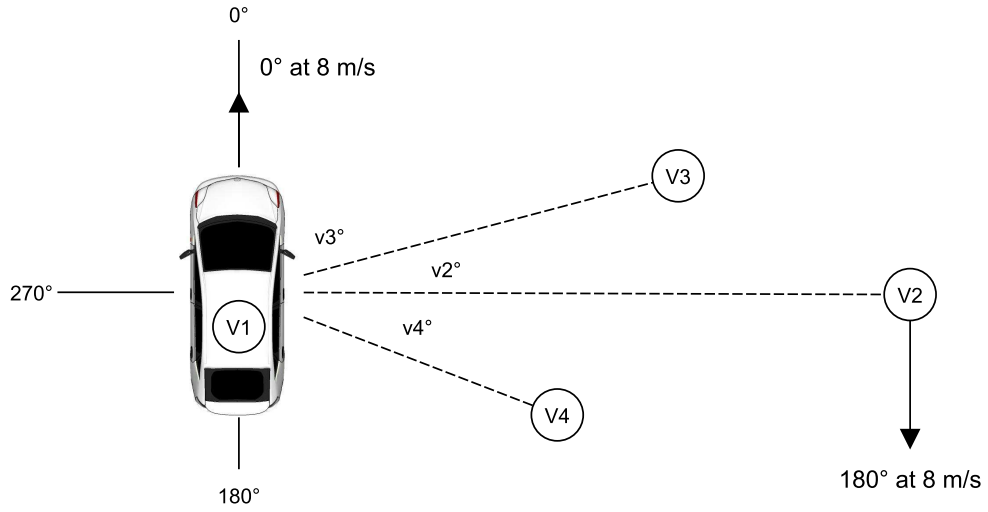


Figure 3.4.: Vehicles V1 and V2 travelling in opposite but parallel directions, within a measurable range of one another.

The speed of a vehicle represents its physical speed in relation to its previous positioning (for example meters per second).

Figure 3.4 shows an example scene containing four vehicles. Vehicle V1 is within proximity of V2, V3 and V4 and vice versa. Each vehicle holds a differing relative measurement concerning their position, bearing and speed to one another. A bearing to the north compass is used as a relative positioning method for multiple vehicles. In the example, vehicle V1 and vehicle V2 are both travelling in complementary directions to one another (V1 on a 0° bearing and V2 on a 180° bearing). We know that V2 is travelling at a speed of 8 meters per second. If we combine the mobility data of both, we are able to calculate relative proximity and orientation data, as well as the relative angles between vehicles (V2, V3 and V4).

The range between V1 and V2 represents the distance (D) between the vehicles. Within a minimum distance vehicles are said to be within proximity of one another. At this particular time instant, V1 sees B at a 90° angle and V2 sees V1 at a 90° to itself, where each vehicle direction represents its bearing axis. As the mobility model is geodetic, mobility specifications contain a high level of granularity. This granularity, refers to the accuracy between calculating the distances, bearings and speeds of objects when using geographic data. For instance calculating D using the geodetic Vincenty formulae [Vin75] results in a maximum error of 0.5 millimetres given the Earth's curvature. Within the framework, we assume vehicles are capa-

ble of accessing vehicular data, either directly in the form of vehicle state or indirectly using sequences of trace history fragments. Combining mobility data with sensory data constructs increasingly rich contextual information about a scene or location. Vehicles are hence provided a means of determining the state of the environment within which they exist (allowing for error).

Messaging

Within the model, vehicles and static objects broadcast messages. The framework models the V2X IEEE 802.11p standard WAVE Short Messages (WSMs). WSMs are short and broadcast-based, well suited to decentralised vehicular protocols. WSMs are provided as a standard message packet type along with IPv6 capabilities within the IEEE 802.11p standard [IEE10]. At the base level, inter-vehicle messaging within the model is asynchronous and non-blocking. Objects broadcast a message (M) without requiring an acknowledgement (ack) of message receipt. Messages contain service payloads (P) which are protocol specific. Messages received and ready for broadcast are queued in `MessageQueue` data-structures. Messages broadcast are popped off the queue and sent into the world messaging space.

```
1 broadcast(MessageQueue Q){
2       for (int n = 0 ; n < Q.length; n++)
3             world.add(Q.pop());
4 }
```

Broadcasts are either triggered by protocol events, time-outs or a combination of the two. Messaging is achieved over successive time-steps, with messages filtered by geographic position and transmission time-step. Message retrieval is dependent on the broadcasting vehicle's communication range (CR). For example, a M containing P is time-stamped and if sent in t, then we assume vehicles buffer received messages in the next execution time offset $t + k$. Message queues allow a vehicle to store a message from all previous time-steps until the message queue is handled and flushed.

Messages within Q are filtered prior to being provided to a prototype protocol. Most notably, messages must be from a previous time-step and within proximity of the broadcaster. Vehicles filter out messages which they may have sent in a previous time-step (M.bid, where

bid represents the vehicle identity of the message broadcaster).

This filtering and allocation (with the unseen cost of simulating fault) of messages is costly requiring n^2 calculations to determine proximity, added to n calculations for message allocation, added to n calculations for fault emulation - for each cycle of simulation. In a non-mobile context repeated proximity and message allocation calculations are not required, making static systems significantly less complex and less computationally intensive to simulate messaging. From the world perspective, we express the conditional retrieval of a message as the filtering `receive` function.

```
1 MessageQueue receive(MessageQueue Q){
2       for (int n = 0 ; n < Q.length; n++){
3             Message M = Q.get(n);
4             if (proximity(M,CR) && (previousTimeStep(M)) && (M.bid != bid) { }
5             else { Q.remove(n); }
6       }
7       return Q;
8 }
```

An object does not pause the protocol cycle between messaging (a vehicle typically calls `broadcast(Q)` periodically according to the broadcast period). Given a subset of objects within communication range of a broadcaster, a broadcast message is assumed to reach all listening or receiving objects. From the point of view of receiving objects, messages are unicast to them. In reality, broadcasted messages reach only a subset of the vehicles within communication range of the broadcaster due to errors commonly found while transmitting radio data (wireless messaging is modelled as *unreliable*).

We distinguish the limited communication range as a spherical distance from the broadcaster, however in reality it is more likely that this communication range would be irregular (as radio signal propagation effects occur within the surrounding environment). If a neighbouring vehicle exists within the communication range and the message has been received without error or corruption, the neighbouring vehicle is deemed to have correctly received a message. Message fault is modelled as link failure. Figure 3.5 shows an example set of timelined interactions between three vehicles (V1, V2 and V3).

At t, V2 broadcasts a message M1. V1 receives the broadcasted message from V2, while V3 does not as it resides outside the communicable range of V1. At $t+k$, V2 broadcasts a message M3 which is received by both V1 and V3, however a fault occurs in the retrieval of M3 by V3 and the complete message is not retrieved or buffered for processing by $t+2k$. The framework models message faults as link failures.

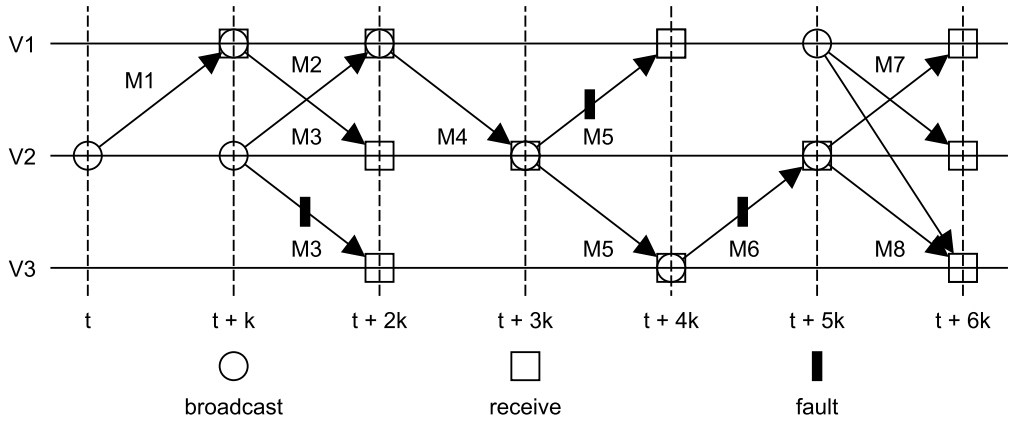


Figure 3.5.: Messaging scenario.

While seemingly trivial, large scale broadcasts present a number of challenges to the protocols that use them. A trade-off of non-acknowledged broadcasting (pure flooding or epidemic routing) is its use of bandwidth. For instance, flooding is typically wasteful in applications where sampled data does not change rapidly and where the capacity of a communication link between nodes is limited (as data requires both storage and communication capacity). Flooded messages travel through the communications network rapidly. The redundancy of data is typically high as data may be duplicated for each successive time-step. The duplication and merging of data requires message and data management at both message retrieval and message broadcast - effectively filtering stored and forwarded data. Redundancy is beneficial to an extent when dealing with fault and erroneous data (data integrity). Within wireless feedback driven applications, such as those described in later chapters, flooding is beneficial as it updates the most recent state of each vehicle to each other vehicle. Vehicles 'eavesdrop' on potentially useful information which is then incorporated or dismissed. A harmful effect of redundancy is that it typically uses large amounts of resources.

For example, bandwidth and storage management schemes are required. The mobility of vehicles and their intermittent contact with multiple contacts complicates management even further as sub-branches of a vehicular network are likely to change. A significant problem and inefficiency is that messages and data can recycle (loop back). For example a vehicle V1 may broadcast data at t. Vehicle V2 may retrieve a payload from V1 and rebroadcast a subset of t-data back to V1 at a future time instance. The need for cyclic data management is dependent on the protocol application goal and the developers intended use of redundancy. Figure 3.6 shows the wrapper `Message` structure, which uses both hop lists and created time time-stamps to manage message redundancy.

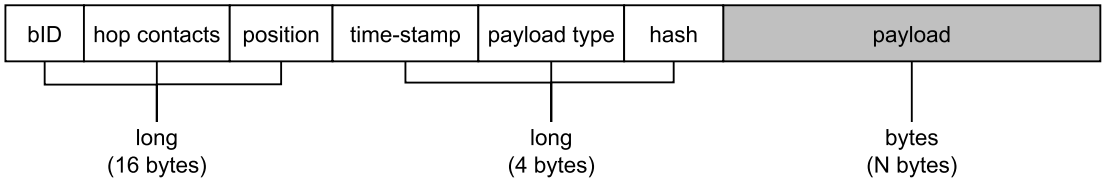


Figure 3.6.: Message structure.

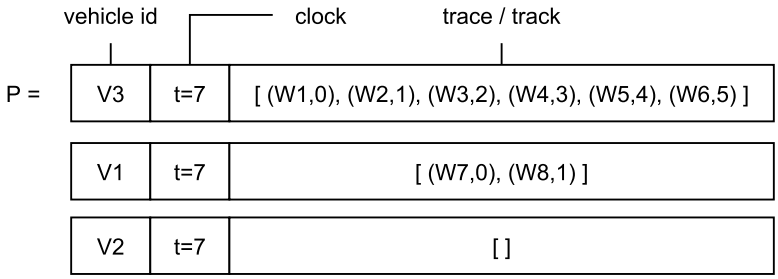


Figure 3.7.: Example Payload.

Each Message structure wraps the protocol specific Payload and holds at least a 60 byte header to identify and filter messages on retrieval. A selection of fields identify each message broadcaster (bID), hop contacts, Waypoint position and created time time-stamp. The hop contacts field is used to identify the occurrence of looping redundant messages. Both hop contact and time-stamp fields provide a means of managing message redundancy. IEEE 802.11p uses a set 1400 byte packet size for WSM data packets. Hence Payload data sizes are limited to a maximum size of 1340 bytes. A hash is used for message integrity. The data held within

a Payload (P) is general. Within the services presented in later chapters, P is specified to hold mobility fragments, which are interpreted heuristically using the function library to abstract more complex contextual information about the road network and neighbouring vehicles. An example payload with mobility fragments contains an ordered set of waypoint-time pairs (W,t) at $t = 7$.

In the example, we are provided a set of payloads (Figure 3.7). A message M from V3 at time 7 is received containing the traces from waypoints $W1$ to $W6$. Other messages are received from V1 and V2. V2 shares no track mobility data. Similarly, payloads may contain data which reports sensing or actuation at particular time-steps. The number of groups formed by vehicles due to proximity are potentially exponential of $O(2^n)$ cost and therefore highly computationally intensive.

Feedback, Protocols and Applications

The architecture (Figure 3.2) leverages feedback [AM08]. We call systems which include human decision making in the feedback loop, *advisory or recommendation systems*. A trade-off of feedback is that it adds significant complexity, not only because protocols are parallel between vehicles, but also because inter-relating feedback loops change at successive time-steps due to mobility. As vehicles travel they form multiple feedback loops with one another, both physically through the environment and via messaging (Figure 3.8). The structure of these feedback loops and systems change, consequently dynamism makes formalising these changes challenging.

Within decentralised services, feedback is leveraged to adapt service operation. Services which continuously update or refine themselves to maintain a measured operation are well suited for decentralised services. However, feedback loops require developers to think slightly differently about a problem. Notably, a service needs to be written while considering the often gradual adaptations vehicles must make to achieve a state. As time progresses services should attempt to adapt themselves towards a goal state. Specifying this goal state often requires a utility metric to calculate the error between the current state and the goal state. Figure 3.8 shows the feedback loop for a single vehicle, assuming that a driver follows the protocol (Pr). A starting state X results, after processing, as a resultant state Y.

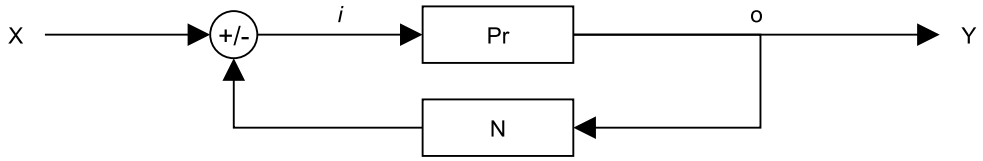


Figure 3.8.: Simplified data feedback loop as a block diagram.

The set of inputs (i) include the clock (t) as well as a summation of messages (m*) by neighbouring vehicles (N). Each Pr is furthermore provided a unique geographic position (p) for each vehicle. Therefore i = (t, m*, p). We write the simplified vehicle feedback loop as:

$$\frac{Y}{X} = \frac{Pr}{1 + NPr} = V$$

The complexity of parallelism is simplified here using N. However, N and the structure of feedback parallelism is dependent on the proximity of vehicles to one another (which is measured geographically and measured against the communication range). Mobility affects the structure of feedback and the connectivity between feedback loops. Where multiple vehicles inter-act, we require the expansion of N of the form $V_1 + V_2 + V_3 + ... + V_n$ for n-many vehicles (Figure 3.9).

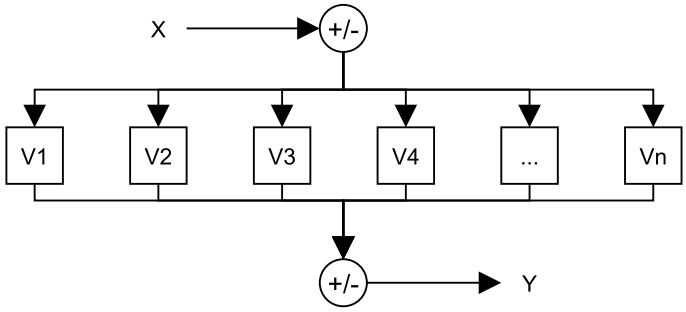


Figure 3.9.: Parallel feedback.

To determine whether feedback adaptation should occur a vehicle requires a data store and a set of conditions to compare previous states with the present state and thereby choose the appropriate actions necessary to optimise service. Adaptation is a response which seeks to optimise vehicular behaviour to improve either individual or group performance. Protocols

attempt to leverage both messaging and mobility which exist in the vehicular architecture. Feedback is however dependent on the ability of the vehicle or object to store state data from previous time-steps. A service should seek to continuously adapt towards equilibrium service operation. In this example, a metric value of 1.0 represents true equilibrium, removing the need for further adaptation (Figure 3.10).

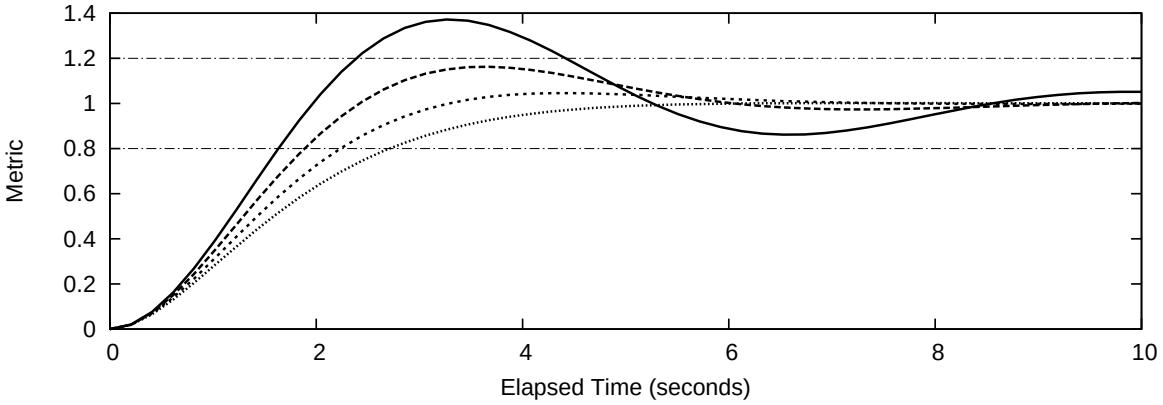


Figure 3.10.: Services ideally adapt towards equilibrium performance over time.

Decentralised vehicular services are comprised of *protocols* (Pr), *applications* and *payloads*. Within the framework, services are assumed homogeneous between vehicles. In other words, vehicles all use the same service components; however, it is possible to have more than a single derivative of a protocol operating simultaneously. Each service requires at least one protocol. Protocols specify the sequence of procedures executed when receiving both sensory data and/or messaging data - determining how to handle message inputs in combination with positioning and time information. Protocols are responsible for interpreting retrieved messages, interpreting sensory data, monitoring the local clock, executing the specified protocol, constructing new broadcast messages for the next time-step and if granted access - applying actuation using actuators of the vehicle (Figure 3.2). Multiple protocols are capable of running simultaneously providing differing or complementary vehicular services. However, conflicting services should be avoided, for example those services which may produce race conditions or contention of resources between one another.

```
MessageQueue Protocol(MessageQueue inbox, long t, Waypoint p, Store s){
      //Filtering
      ...
      //Processing
      ...
      //Finalisation
      if (condition(t,p,s) == 0){
            Payload p = new Payload();
            Message msg = new Message(p);
            ...
            outbox.push(msg);
      }
      return outbox;
}
```

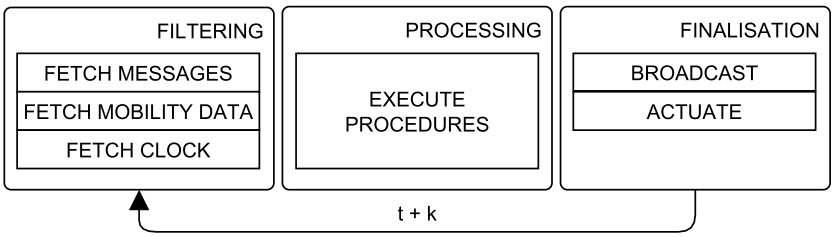


Figure 3.11.: Procedural transitions for each protocol loop.

Protocols are defined in terms of state transitions and procedures. Each protocol developed uses a protocol template, allowing the developer to focus on messages and sensory data coming in, as well as output messages and actuations. On the cycling of the clock (`t`), a time-out occurs, forcing the periodic execution of the protocol.

The specification of the protocol loop is broken into three phases: (a) *filtering*, (b) *processing* and (c) *finalisation* (Figure 3.11). For each `Message` retrieved by a vehicle, the `Payload` is identified using the payload type identifier and queued for processing. Payloads are typically not in-order and reflect only the previous time interval. The processing phase applies the actions of the protocol - we explore specific protocols in later chapters. The processing section uses interfaces to access sensory data via the function library (Table 3.1) and a local bounded store. Processing results in the creation and buffering of new messages for later broadcast.

The combination of control and messaging allows sensory inputs and messages to influence

control as well as to affect change using communication and actuation in a future time-step, while considering the various existing feedback loops (Figure 3.2). Applications within the framework do not modify the operation of a protocol; rather they observe data streaming through the protocol and thereby provide an interface between the protocol and the driver concerning the state of a service. The driver in turn can facilitate changes to the operation of a vehicle which thereby effecting the data or state provided to the protocol. The operation of the service is subsequently dictated by the dependencies existing not only between a vehicle and its neighbours, but with the previous interactions of itself and its neighbours in all past interactions. Hence the actions taken by a vehicle are a summation of previous states and actions in previous time-steps. We call this linking of feedback loops - *dependency*.

Dependencies

To observe service operation during development, developers have a number of options. They may either observe the behaviour of the system using visualisation methods and metrics and/or they may interpret behaviour from recorded actions. The dependencies existing between vehicles reference the reactive adaptation vehicles take in response to changes in their environment. They provide one of many possible means by which we may review service behaviour in a graphical form. An example dependency can be considered, where a vehicle V2 is following V1 with a following distance. If V1 brakes, then V2 is required to adapt to change made by V12 to ensure the following distance is corrected (and equilibrium is maintained). V2 is thus dependent on V1, but V1 is not dependent on V2.

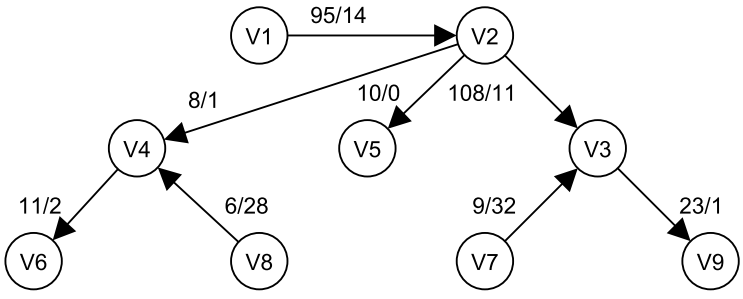


Figure 3.12.: Dependency: vehicles communicate with one another in various time intervals. The actions made by one vehicle effect the actions of another vehicle through feedback. We represent this effect using directed arrows and weighting. In this example, V2 is dependent on V1. For example 95/14 depicts the 95 adaptations which V1 imposed on V2 and the 14 adaptations imposed by V2 on V1.

Within the framework, dependencies can be graphed for given time intervals associated with contacts. Figure 3.12 shows an example of the dependencies existing between a collection of vehicles (V1 to V9). Within feedback-based protocols vehicles may alter their data or actions based on previous interactions between vehicles they may not have themselves come into contact with. In the dependency example V1 influences both the sub-graph of V2 contacts and V2 itself. Dependencies typically make protocols challenging to build and introduce complexities in protocol design. Understanding dependency serves to determine when a vehicle should broadcast messages. Network analysis provides methods by which to interpret dependency graphs and thereby understand visualise the effects of a particular protocol between vehicles.

3.2.4. Function Library

The function library within the framework derives complex data from position (spatial) and time (temporal) data primitives. Various geographic algorithms [Rog07, Vin75] are used to calculate position, distance and bearing using waypoints. Using heuristics and linking functions, data is used to abstract contextual information about the state of the environment in which a vehicle resides. Functions allow us to refer to geographic locations in terms of sectors and regions. For example, a Message may exist 'ahead' of a vehicle. We might determine a vehicles position within proximity to other waypoints. Table 3.1 and Appendix A.5 show our primary set of library functions, however further functions can be derived to extend the capabilities available to a service. The function library provides a toolbox of off-the-shelf algorithms.

From primitive `position` and `time` (*) functions we derive a number of complex functions. However, there are nuances in the precision and accuracy of particular functions, as well as assumptions made by those functions. An example of this problem is estimating the precise future track of a vehicle. Future track is challenging and is itself a separate problem of estimation and prediction. Within the framework, the future track is inferred using a given route (R). A vehicle uses the route (R) as a plan and combines the route with an average speed (S) to calculate the interim positions given execution cycle length. The plan provided is hence expanded to associate a set of time instances with a set of estimated future positions. In effect, the granularity of the plan is improved to allow vehicles to predict where a vehicle will be located along the road network in T future time steps.

A problem with estimating vehicle mobility in the future is that such estimations are likely

Function	Description
`time()*`	Return the present clock time (t).
`position()*`	Return the geographic waypoint (W) of the vehicle in the present time instant.
`bearing()`	Returns the relative angle to 0° given the direction of travel between geographic positions in the previous time instance and the present time instance.
`speed()`	Return the instantaneous speed of the vehicle using the waypoint from the previous time and the present time instance.
`distance(`X`,`Y`)`	Calculate and return the Vincenty [Vin75] distance (in meters) between waypoints X and Y[1].
`bearingBetween(`$W1$`,`$W2$`)`	Return the smallest angular difference (θ) between two waypoints $W1$ and $W2$ from the viewpoint of $W1$ and the bearing of the determining vehicle.
`isAhead(`M`,`θ`)`	Return a boolean result determining whether a broadcasting vehicle (broadcasting M) exists within an angle of θ degrees of the receiver.
`proximity(`M`,`D`)`	Return a boolean result determining whether another broadcasting vehicle (broadcasting M) is within a minimum distance (D) of the receiver.
`getClosestMessage([`M`])`	Return the message (M) which was, in the previous time-step, the closest geographically.
`destination(`W`,`B`,`D`)`	Return the waypoint destination given a present position (W), a bearing (B) and distance (D).
`nextPosition(`W`,`B`,`S`)`	Return the predicted next waypoint of a vehicle given a starting position (W), a bearing (B) and speed (S).
`getTrace(`T`)`	Return the ordered set of waypoint-time pairs for the trace history limited to the last T time-steps.
`getRoute()`	Return the ordered set of waypoints which specify the route (R) for a vehicle.
`getFutureTrack(`R`,`S`,`T`)`	Return the ordered set of waypoint-time pairs for the future track limited to the next T time-steps for a given route (R) at a given speed (S).
`isFollowing(`M`)`	Ascertain if the message broadcaster of message (M) is being followed by the receiver. If a broadcaster has similar bearing and is ahead of the receiver, then the receiver returns true.
`meet(`P`,`Q`,`D`)`	Given two future tracks, P and Q, estimate the time-step when the owner of future track P will come within proximity (D) of the owner of future track Q.

Table 3.1.: Primitive and derived library functions.

to change due to events and dependencies. However the cost of estimation is less than a brute force approach as vehicles are constrained to travel on roads. Long range time scaled position predictions are typically more inaccurate than short time scales. Vehicles are dependent on the interactions occurring between a vehicle and its neighbours and the road infrastructure in future time steps, which often may not be known. Hence the longer ahead a vehicle estimates future mobility the more likely that mobility is erroneous. Future track is estimated by expanding the intended route of a vehicle. The route is converted into a fine grained mobility plan. The more long range an estimation in terms of time or distance, the more resources are necessary. The `meet` function depends on the assumptions and quality of estimations made by `getFutureTrack` algorithm.

Functions which abstract the complexities of spatial and temporal references are beneficial such that they simplify the manner in which a protocol considers scenarios which are influenced by position and time. For example, determining if a vehicle is `ahead` of another vehicle removes the consideration of intermediary primitives to calculate this position information.

Human Drivers

The services constructed with the framework are considered *advisory* services. Such services enhance the data available to a driver or community. Vehicle *drivers* represent the highest component in the architecture, interpreting observations of the physical environment and data provided by the service application. A driver's response to these recommendations and advice may result in an actuation, similar to that produced by the protocol loop in an automated setting. Without the presence of a protocol loop and the application as a service, a driver is limited to just physical observation of the road scene. This limits the driver, as messaging can typically retrieve positioning data from objects beyond the sensory range of a driver. In the presence of an application a driver is typically provided an enhanced view of the world space - filtering both physical data from the scene and information provided by the service application.

Hence, a driver is modelled in our framework as a filter to input data. A driving filter value of 0 corresponds to no actuation on the part of the driver, whilst a filter value of 1.0 corresponds to actuation on all received inputs. We assume that the driver has a minimal response time when dealing with both application alerts (provided by a user interface) and physical observations. While major improvements have been made to driver-less vehicles [Thr10], the model seeks to

share responsibility of vehicle routing with the driver in situations where the protocols may need modification. The legal ramifications of fully automated driving have yet to be clearly specified. For instance, in the event that a vehicle needs to complete a U-turn, a driver is given responsibility of taking over the task - after this has been completed the vehicle may re-engage usage of an automatic-drive system. Similarly, while vehicles in the future may become driverless, in the meantime vehicles will have drivers, who will have varying responsibility for driving the vehicle.

3.3. Simulation

In an ideal development environment, a simulation tool would be consulted as a means to accurately model (without error) how a system would behave if deployed. However perfectly realistic simulation is computationally infeasible. Also, attaining the ideal simulation environment is challenging particularly as modelling real world processes is also itself challenging. For example, simulators for weather prediction (considered a complex system) use large clusters of high performance computers to model a variety of atmospheric and geographical components, where regions interact (communicate) with one another.

Even given these capabilities, weather prediction is an inexact science. To balance these trade-offs, simulators typically select the essential features of models and systems to approximate performance. This adds to the challenges of simulation. Different simulators focus on differing measures and aspects of systems operation, some contain similar features but model systems uniquely. In the following section we describe the Geographic Urban Simulator (GUS), its architecture, usage of Java in Simulation Time (JiST) and operation.

3.3.1. Java in Simulation Time

JiST was initially provided for building Discrete Event Simulators, termed *virtual machine-based simulation*. In comparison to other simulation frameworks, JiST is highly efficient, outperforming many optimised simulation run-times both in execution time and memory consumption [BHvR05a, BHvR05b]. It also inherits a number of features from the Java language including reflection, type-safety and garbage collection characteristics. The JiST architecture is comprised of four components: a *compiler*, a *bytecode rewriter*, a *simulation kernel* and a

virtual machine. In a simulation, JiST converts an existing virtual machine into a simulation platform, by embedding simulation time semantics at the byte-code level. Simulations are written in Java, compiled using a regular Java compiler, and run over a standard, unmodified virtual machine. The process is illustrated in Figure 3.13. JiST allows us to construct large scale simulations.

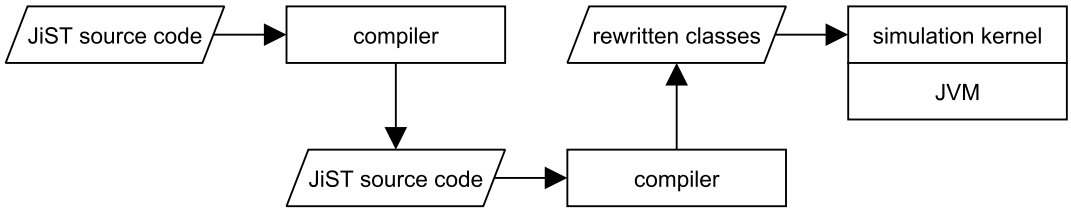


Figure 3.13.: Code/JiST compilation/rewriter process [BHvR05a].

3.3.2. Geographic Urban Simulator

The Geographic Urban Simulator (GUS) is a modular JiST extension for the simulation of large scale decentralised vehicular services. GUS implements the decentralised vehicular framework within a simulation environment. Appendix A provides more detail on the GUS architecture and its performance including code metrics and object hierarchies.

GUS focuses on providing a number of mobility and communication features that are not easily applied or available using other simulators, namely (a) *control emulation*, (b) *geographic positioning*, (c) *dynamic mobility* and (d) *short messaging*. The code of a decentralised service (protocols and applications) are written as if being deployed on real devices. In contrast to the scripting approaches used in other simulation frameworks [MI04, Cha99, HJ08], decentralised services are written in Java as they would exist if deployed on devices (e.g. Android Dalvik devices). The developer only needs to concentrate on the `Payload`, `Protocol` and `Application` components of the service. Short message payloads in the GUS use the WAVE Short Message standard. A direct trade-off of increased `Payload`, `Protocol` and `Application` realism is a reduction in the performance of GUS. The complexity of the `Protocol` and `Application` increases the resources used by the GUS. Increasing the number of simulated vehicles in the system also increases time costs.

The core of simulator consists of three layers: (a) the Java Virtual Machine (JVM), (b) the JiST framework, (c) the GUS modules. Figure 3.14 illustrates this layered architecture. GUS

uses user parameters, a service protocol, payload description and mobility traces to simulate a given scenario.

Before a simulation, GUS requires a set of mobility patterns with which mobility can be specified for a given population of vehicles. Mobility patterns are derived either from real traces or from synthetic traces. A converter allows the GUS to use traces from a variety of mobility sources [NBG06a, Goo11, BGJL06]. These sources are interpreted as sets of mobility specifications. The GUS mobility specification is used to describe the geographic route and movement of a vehicle. Below is an example specification for a straight line route from A to B given in the form [(A):S:(B)].

$$\begin{bmatrix} (32.263, -114.572) : 8.0 : (32.262, -114.573) \\ (32.262, -114.573) : 6.2 : (32.291, -114.588) \\ \cdots \\ (32.232, -114.324) : 5.0 : (32.277, -114.589) \end{bmatrix}$$

The sequence states the start position (A) of a vehicle at the first latitude-longitude pair (32.263,-114.572). That it may travel at a maximum speed of 8.0 meters per second towards (B) at (32.262,-114.573). Extended and linked sequences are capable of producing complex routes through a road network. Importantly, specifications are themselves interpreted as a plan rather than a fixed routing of a vehicle from A to B. The plan can be modified in the form above, such that a vehicle service may dynamically route itself through the road network. This allows modelled vehicles to apply actuation, for example brake, accelerate or turn.

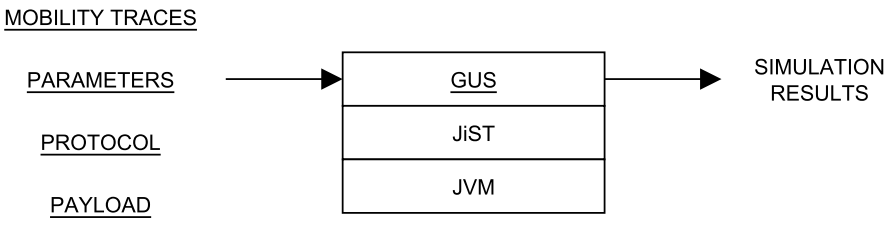


Figure 3.14.: GUS Simulation process.

A decentralised service is the combination of Payload and Protocol components. An Application component is optional as an interface to the Protocol. Parameters provided to the simulator include the number of vehicles for a given experiment, elapsed experiment times,

default communication range, dropped packet percentage, as well as other values to be used by the service. For each simulation, a log of results is created that can be interpreted and analysed either visually or using a variety of metrics (including message counts, drop message counts, contacts seen, the position of interactions, the number of vehicles which reached their destination successfully, service specific metrics etc.). Metrics can be split into physical and networking metrics. Physical metrics consider calculations within the physical environment of a vehicle, for example, distance calculations. Networking metrics consider communications metrics, for example, packet size, message delay, message throughput. Measurements are recorded every 100 milliseconds and adds to the computational load placed on the simulator. We still expect service developers to test the operation of protocols on real test-beds and devices.

GUS uses a collection of helper applications to interpret and visualise data, including GraphML[2] and GNUplot[3]. Due to a significant lack of mobility traces, the default mobility generator used with the GUS, scrapes public navigation directions from Google Maps [Goo11] and Open Street Map [Ope11]. By specifying two geographic positions (a start position and an end position) and a range distance for a particular zone, the generator chooses a number of random locations within the zone. These random locations represent the origins and destinations for vehicles. Querying navigation websites provides driving directions in the Keyhole Mark-up Language (KML) format. KML directions are interpreted and converted into GUS's mobility specification format previously discussed. KML directions are used as they associate a significant amount of meta-data with road sections (including speed limits and expected road section travel times). Such maps present a good source for data as the routings provided are influenced by meta-data, for example road width and speed limit data.

3.3.3. Performance Comparison

GUS simulation performance is largely dependent on the hardware on which the simulator is executed. Figure 3.15 presents benchmark tests for messaging protocols which do not use decentralised internal data structures for unique vehicle storage. For example, it is possible for a protocol to implement and manage its own unique data stores during simulation. Benchmarks were conducted on a single Intel Core 2 Duo 2.80 GHz machine with 3.72 GB of usable memory.

[2] `http://graphml.graphdrawing.org`
[3] `http://www.gnuplot.info`

Reading the benchmark mappings, from bottom to top, we can ascertain the time required to simulate scaled populations of vehicles. Notably, the benchmarks provided do not consider the running on clusters of machines.

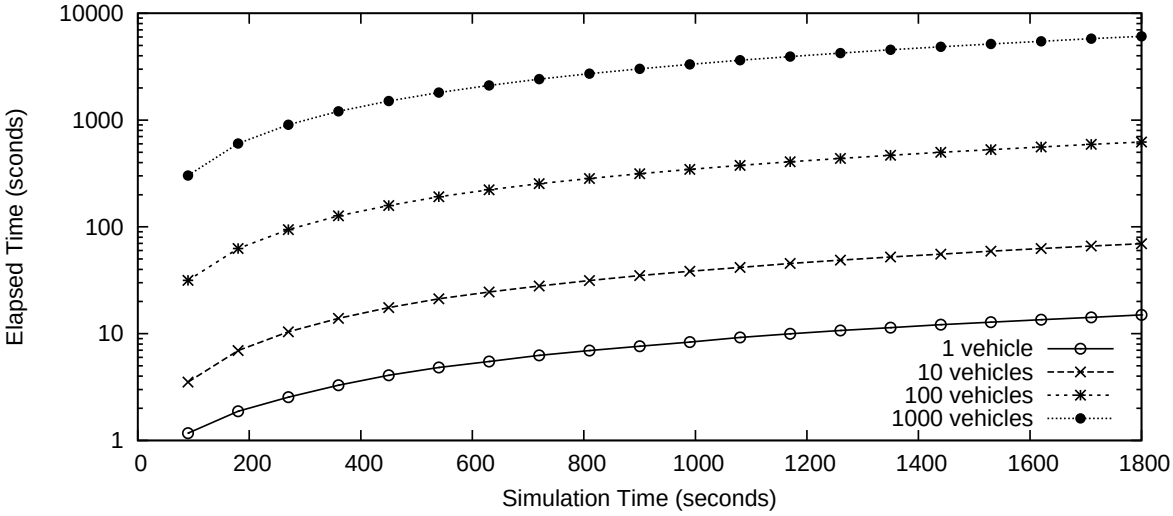


Figure 3.15.: GUS performance: elapsed experiment time versus simulation time.

JiST has been compared in previous work to NS-2 [KS07]. In comparison to previous approaches, GUS simulation performance was better than NS-2 but worse than GLOMOSIM and SWANS (Table 3.2). The GUS is directly motivated by the successes achieved through the composition of JiST and SWANS [BHvR05b]. JiST/SWANS presents a high performance and lightweight VANET simulation environment in which Java-based message routing protocols can be written and evaluated. In terms of modelled communication layers, SWANS models the OSI stack to mimic message transmission in VANET scenarios. However, SWANS has drawbacks. Notably, SWANS does not support dynamic or changing mobility. That is, vehicles cannot modify their routing at simulation time. SWANS also does not support the implementation or swapping of protocol loops. Simulation statistics and metrics are not typically output, unless explicitly called. The ultimate aim of modelling and simulation using GUS is to achieve a best effort measure which approaches as close as possible the performance of a real world deployment of a decentralised vehicular service.

Lower performance in GUS is partly due to the complexity of the simulation components (e.g. `Protocol`, `Payload` and `Peer` objects). As a trade-off of enabling prototyping, performance was seen to decline for larger scale systems. On a single processor, GUS is capable of simulating

Nodes	*NS2*	*GLOMOSIM*	*SWANS*	*GUS*
100	1070.4s	12.6s	6.4s	221.32s
1000	-	948.6s	67.8s	2111.9s
10000	-	-	683.8s	24538.2s

Table 3.2.: Simulation performance comparison.

systems much larger than 10,000 nodes. As vehicles are heterogeneous in state and complex in their operation, increased resources are used.

3.3.4. Discussion

Simulation is itself a challenging problem. As computational capabilities are limited, simulator designers need to make choices about which features to implement in a given simulator. The more features or realism found within the simulation model, the larger the resource requirement. Hence, design choices are made while considering the trade-offs of given approaches. GUS specifically allows us to rapidly swap and prototype protocols in Java (a commonly used language). In particular, GUS allows dynamic changes to model mobility and protocol properties at simulation time. Mobility traces can be input from a variety of sources (both synthetic and real). Synthetic trace generation tools which leverage mapping services like OSM [Ope11] and Google Maps [Goo11] allow us to consider different mobility maps for different cities. In contrast to other approaches, protocols are written as prototypes, in order to achieve a balance between levels of abstraction and implementation.

GUS leverages a relational database for the collection of simulation results and metrics. A collection of helping tools allow a protocol designer to analyse the performance of protocols. As a trade-off of providing these features, the GUS makes assumptions concerning the realistic transmission of messages within urban environments. GUS messaging does not accurately consider the network and physical layers occurring in other OSI based simulators. Noise (e.g. multi-path reflections, attenuation, etc.) and disruptions which might effect a radio signal in an urban setting are considered to challenging and computationally intensive to model. Rather, GUS drops complete messages to simulate packet loss and message fault.

An issue with the GUS is that it is tailored specifically to evaluate decentralised vehicular service protocols, making implementing and comparing protocols with other architectures difficult and prone to errors and differing semantics. A solution to this is to adequately and

accurately present comparable components and scenarios to various simulators as standard benchmark scenarios. These benchmark scenarios do not exist in the literature. By repeatedly simulating scenarios and monitoring their results, simulation provides the framework a means of improving confidence in a service's behaviour. Indeed, if simulations run for a long time without error we improve our confidence in the approach further.

3.4. Summary

In this chapter we began by detailing the requirements of a decentralised vehicular framework. We described our vehicular simulation framework for the construction of decentralised, scalable broadcast-based, feedback-driven, vehicular services. The framework provides a scaffolding on which decentralised vehicular services are prototyped. In contrast to other approaches, mobility within the framework is considered fine grained, highly accurate, flexible and shareable. Vehicles construct services through sensing, inter-vehicle messaging and the sharing of mobility fragments. Increasingly complex geodetic methods and functions are derived and used to calculate the distances, angles and relative speeds of objects. While seemingly simple, scaled services present challenges to developers as vehicles rely on mobility and feedback loops to construct decentralised services in a dynamic and ever changing environment.

The Geographic Urban Simulator (GUS) is used to simulate services developed within the framework. Simulation provides tools and means by which we reason about and evaluate protocols and decentralised services, thereby improving our confidence in the feasibility and operational behaviour of the service prior to deployment. The framework provides a method of abstracting away the complexities of service design allowing developers to focus on the Protocol, Payload and Application components of a service. The Java in Simulation Time (JiST) framework was briefly described in conjunction with the features embedded within the GUS. We concluded by comparing and critiquing GUS in comparison with other simulation methods (NS-2, GLOMOSIM and SWANS).

The following chapters elaborate on the framework using two case studies, namely travel time estimation and intersection control. Centralised versions of these services are complex and costly. Using our framework we develop, evaluate and compare decentralised solutions.

4. Travel Time Estimation

In the previous chapter we detailed our geodetic based framework and simulation framework. The framework presents us with a method for developing and simulating large-scale decentralised vehicular services. To assess the practicality and performance of the framework, this chapter develops a decentralised travel time estimation service. The service is non-safety critical and while seemingly trivial is complicated by the dynamism and scale of road networks. In the next chapter, we consider the problem of intersection control, which is a safety-critical service and operates over significantly shorter periods of time. Using mobility fragments and their derived travel times, the service presented collects and shares recorded travel times. In turn collected and disseminated travel times are integrated to model traffic patterns on the road network. Given the availability of such mappings, vehicles can make informed estimates about the travel times required to travel from a present position to an end position. Shared trace fragments improve the accuracy of travel time estimations as they enhance estimation methods. In contrast to methods which use inductive loop detectors [Kri08, Rob05], our travel time estimation service is decentralised scalable and feedback driven.

We begin the chapter by detailing the problem of travel time estimation (Section 4.1). We highlight the need to determine the time required for vehicles to discover a minimal dataset for estimation (i.e. their availability in comparison to centralised approaches) and the capacity needed by a decentralised approach to improve on a centralised vehicular service in terms of performance. Travel times are defined in relation to the road network. Metrics used are defined in Section 4.2. Section 4.3 describes the service scenario. The mobility traces simulated are sourced from two sources: (a) driving directions services and (b) the ETH trace data-set. We discuss the mobility traces used in Section 4.4. The service protocol we use (Collect-Merge-Share protocol) is presented in Section 4.5 and evaluated in Section 4.7. We use four city maps to validate results and interpret the affect of differing mobility patterns. Before concluding, we

discuss the advantages and disadvantages of the approach as well as future extensions of the application (Section 4.8).

4.1. Problem

Travel time estimation refers to the process of *estimating the elapsed time required to travel from a start position to an end position* using a road network [Kri08]. Commonly this measure is referred to as the *estimated time of arrival* (ETA). Estimation is typically used in vehicle routing algorithms as a means by which to predict route performance. To accurately estimate the travel time between two geographic positions, algorithms consider both travel times and contextual data about the state of the road network. For centralised approaches, accurately estimating the time to arrival on road networks is highly challenging partly because road networks are both very dynamic and monitoring dynamic changes in the network is difficult (requiring repeated sensing). By interpreting the travel times collected by a community of vehicles, such data can be used to estimate the flows and delays occurring across a road network.

Many commonly found in-car navigation devices already provide drivers with ETA information as they drive. However, ETA algorithms used to calculate such estimations often use just two *complete* data sources: road maps and speed limits (*static data*). Maps are assumed complete such that roads and speed limits existing in reality exist within a dataset and are not prone to change significantly over time. Estimations do not incorporate *rich* or dynamic contextual meta-data about the road network (*dynamic data*). Travel time data allows a vehicle to periodically re-evaluate the current state of the road network and heuristically infer the state of the road network in time. The decentralisation of this task is well suited to vehicles supporting V2X technologies. Furthermore, travel estimations and their mappings present a foundational layer on which higher level vehicular services such a vehicular re-routing strategies can be applied. The travel time estimation service considered within this chapter is a decentralised service and uses: (a) road maps and (b) *incomplete* shared travel time overlays from collections of vehicles and (c) varying sample times.

Given a road network, a challenging question to answer at any time is: "What is the *best* route from the current position to a given destination?" This query is regularly made to centralised mapping services. Firstly, the criteria for 'best' route is largely dependent on the criteria chosen

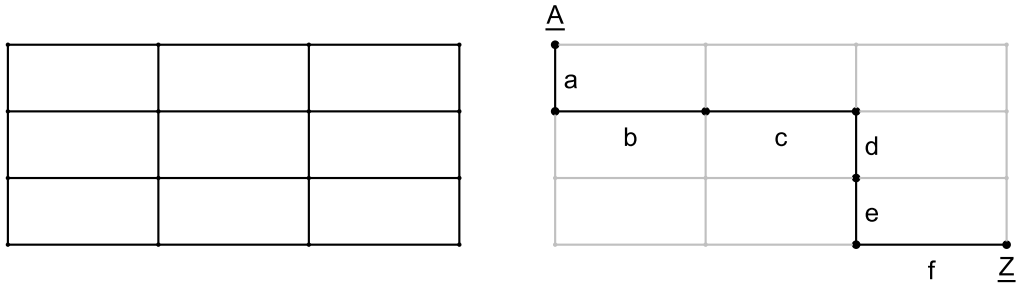


Figure 4.1.: A simple example of a road network (left) and one of several routes from A to Z (right).

by the user. Such criteria may refer to a route with shortest path time, least number of turns etc. Secondly, the time when a query is made may affect the route and estimation provided. As road networks are dynamic, some road sections may perform well at off peak times, but not at peak times. Estimations provided are dependent on the quality of the datasets used to calculate an estimate. We consider *travel time estimation* as the process of predicting, within error, the elapsed time required to travel from one geographic position (A) to another (Z) using a transportation network and a specific subset of the road network. Estimations provide the driver or navigation system with a best effort means by which to differentiate sets of routes. Thus far, most estimation approaches have considered limited datasets in their prediction of travel times. While a number of routing algorithms already exist, many algorithms rely on graph datasets as a means of representing the costs of using specific road sections in their routes.

Road networks are often represented as weighted directed graphs (Figure 4.1). A directed road network, $G = (V, E)$, contains a set of road intersections (vertices, V) and a set of connecting road sections (edges, E). While the structure of the road network or graph remains largely static over time, the associated state and loading of road sections tends to change over time. Hence the associated contextual weights fluctuate or change as time progresses. One example solution is provided by the ordered set of intermediary road sections (a,b,c,d,e,f), which is a subset of E. Given methods of calculation we can estimate the travel time between two geographic positions using speed, time and distance formulas. There are some limitations of this estimation. An estimation made using this method assumes that only a single vehicle exists on the road network in time and that this vehicle is not influenced by any other aspects

of the road network. In reality, vehicles are influenced and effected by a large number of factors (some not easily measured or predicted) including weather, road maintenance, vehicle congestion etc. To improve the quality of an estimation we require rich contextual datasets. This contextual data serves to improve the estimation models used. The type of contextual data available to a vehicle is limited by the sensory capabilities of the vehicle. Vehicles are capable of monitoring their own geographic position using GPS and timing their travel using an on-board clock. Hence, vehicles can record the experienced travel time required to travel a road section.

Even if we are able to accurately sense the state of the road network, we are presented with a further problem, namely that roads are dynamic. The road network is constantly changing and travel times which have been collected may not be applicable to the current or future state of the road network or be up-to-date. Effectively, the travel times recorded lose their value as time progresses. Mobility trace data about the paths of vehicles needs to be collected, filtered and analysed repeatedly and continuously to maintain a view of the road network state. Some centralised approaches have sought to collect such contextual data for the benefit of the road network and the reduction of traffic. Unfortunately, practical trade-offs exist in achieving the ideal of complete road network knowledge. Centralised solutions commonly require large sets of costly sensors to collect road data, for example vehicle counters, cameras and induction loops. Implemented approaches have integrated stationary (static) sensors at fixed points throughout a road network to measure the movement of vehicles. While the sensory costs of this approach are continuously declining, such centralised systems entail expensive management, deployment and maintenance costs. An alternative solution is to use vehicles to collect data and upload this data to a central authority for processing [HWH$^+$10]. In effect, vehicles record and submit to an authority, their experienced travel time using a road section. While estimation datasets are improved, a reliance is made on a central authority to provide such a service. Such a central authority remains a single point of failure and may control access to specific travel data.

4.2. Metrics

For a given journey, Estimated Time of Arrival (ETA) represents a predicted future time instant when a vehicle is expected to reach its destination, while the *Actual Time of Arrival* (ATA)

represents the time recorded by a vehicle reaching its destination (derived from the finish time). Hence, the accuracy of a calculated ETA at a previous time can be measured by calculating the error (e) between the initial ETA and ATA.

$$e = |\text{ATA} - \text{ETA}|$$

Ideally, the ETA should equal the ATA. In reality, the accuracy of ETA is affected by events and obstacles occurring along the route. Road traffic, congestion, weather, maintenance and other issues affect vehicle journeys and often degrade the accuracy of the ETA. The ATA can be seen alternatively as a measure of the real performance of a series of road sections. Subsequently, we can infer that if a road section is not performing optimally, then we should avoid it if performance is less than an alternative route. Among the questions to consider are: What data to collect? When to share this data? How to manage the data? What data is relevant? What data is not relevant? Is more recent data more applicable than older data?

The challenge of building a vehicular service using feedback is made more manageable by using a phased approach as discussed in Section 3.2.3. The problem of travel time estimation using feedback can be broken into three data challenges: (a) data collection, (b) data sharing and (c) data management. Data collection questions not only what data is relevant for travel time estimation, but also how does it lose its relevance over time. Specifically we measure the time taken to collect or update travel times and the quality of this data. Data sharing is necessary as a vehicle has limited sensor range. For example, a single vehicle cannot drive and sample all road sections simultaneously. The sharing mechanism employed is a one-to-many relationship between vehicles. This broadcast approach to travel times produces multiple copies of data (redundancy). Hence, we require a means by which to manage travel time data. More samples translate into an increased dataset and, depending on the diversity of data, a richer dataset.

Using the framework, travel time estimation, within this context, is dependent on the collection, sharing and management of recorded travel times. The ad-hoc networks formed within the network are often unpredictable. Estimation is calculated using limited knowledge. For instance, estimating the travel time from A to B, we may know nothing else than the physical layout of a road network in the form of a map. Provided we have a route, we can use such

a map and an average speed to predict how long it would take to travel from A to B. This calculation is optimistic (a best possible outcome). While a road network map typically conveys geographic and road rule information, for example speed limits and controls, it lacks the attribution of up-to-date contextual (road state) information. This travel time estimation can be described as the function:

$$\texttt{function(route, map, speed)} \rightarrow \texttt{ETA}$$

An estimation function provided with `route`, `map` and `speed` data, estimates the travel time required. The estimation calculated is compared with the ATA which results in an error deviation (either positively or negatively). This approach is insufficient for significantly larger errors. An extended approach to such an estimation calculation is to use sampled travel times experienced by other vehicles to enhance the accuracy of the estimation calculation. We can consider such a calculation as the function:

$$\texttt{function(route, map, dataset)} \rightarrow \texttt{ETA}$$

This approach to estimation is influenced by the incorporation of sampled travel times in the form of a `dataset`. The inclusion of sampled travel times requires that we collect and share data between vehicles. In this approach, an ideal travel time estimation calculation would have available to it the up-to-date data pertaining to all sections of road within a road network at all present and past times. Approaching this ideal is challenging. To build a dataset containing the contextual attributes of all road sections would require a large scale infrastructure of sensors. A large collection of data would need to be sampled periodically and stored. Centralised solutions to the problem of travel time estimation already exist - however, centralised solutions thus far have been maintenance-intensive and expensive to deploy.

Data collected has typically not been up-to-date and lacking in resolution. Furthermore, such data is typically not automatically fed back to drivers using the road network - rather, control is applied by the traffic authority using road infrastructure, for example traffic lights. The problem of travel estimation within this chapter specifically focuses on the use of vehicles as a divide and conquer strategy to the collection and sharing of travel times for the estimation of travel throughout a road network. Without a centralised system and a standing infrastructure

of sensors, we evaluate whether a decentralised approach can leverage a community of vehicles to achieve the goal of calculating a more accurate estimated travel time using sampled travel times. Vehicles are limited to sharing data via local ad-hoc broadcast and using position, speed and time data.

4.3. Scenario, Aims and Assumptions

The scenario considered is defined in routing research as using a *static scenario* [DSSW09]. Beginning her or his journey, a driver of a vehicle queries the on-board navigational computer for directions to reach a destination. Using a local lookup or via the Internet, the driver is presented with a set of driving directions and route specifying how a vehicle can reach its destination. Included with these directions is an ETA and a displacement estimation. We assume that the route provided represents the estimated shortest time between two positions using a shortest path algorithm [Dij59, DSSW09] and vehicle speed data (road speed limits and vehicle performance data).

A combination of route and directions is used by the driver to reach her or his intended destination (such as seen in the route in Figure 4.1). We assume that drivers follow these directions without modifying their route. The scenario is replicated for each vehicle driving along the road network. Each navigational computer does not estimate the travel times associated with the provided route, rather vehicles estimate their journey time using the experienced sampled travel times collected and shared by the community of vehicles (where available). The protocol developed addresses the following questions:

1. How long does it take for a vehicle to find all travel times *relevant* to itself. Alternatively, how much time is required for a vehicle to estimate its own route from collected travel times? (Section 4.7.4)

2. How long does it take for a vehicle to find all travel times for a sampled road network? (Section 4.7.3)

3. What is the communication overhead of such a service? (Section 4.7.2)

4. What estimations can be made from known travel times? (Section 4.7.4)

5. What estimations can be made about unknown travel times? (Section 4.6.1)

6. Can we produce travel time maps comparable to those provided by related works [WBT+10]? (Section 4.7.5)

7. Can we produce travel time graphs usable for dynamic routing scenarios [DSSW09]? (Section 4.7.5)

We assume that vehicle mobility is constrained to each vehicle's specified route. Vehicles have the capability of finding their own position, measuring the time to travel between sub-positions and communicating wirelessly with neighbouring vehicles using ad-hoc WiFi (IEEE802.11p WAVE standard [BRCMGCRH08]). As vehicles drive their route, broadcast is data collected by themselves or other past contacts. A vehicle, because of its size and energy provision, allows us to assume that each vehicle is capable of providing large data store capacity.

4.4. Mobility Traces

While a number of mobility datasets have recorded road use within cities, their characteristics and type (for example bus, taxi and car networks) makes them inappropriate for use within our application scenario. For instance, bus routes present us with periodic traces, covering only certain routes, yet they do not record a large number of proximity contacts between vehicles. While taxi traces are more random in their routing, they do not exhibit many interactions between other taxis. In other words the datasets for taxis do not consider groups of vehicles, rather individual taxis. Given these two examples, available vehicular datasets do not consider random interactions between vehicles or scale the number of vehicles using a road network within a particular region.

For the purposes of experimentation we used two source datasets, namely: (a) *generated bespoke traces* and (b) traces from the *ETH Realistic Vehicular Traces dataset* [NBG06a]. Bespoke traces were generated using routing directions provided by Google Maps[1] and Open Street Maps[2]. Given a chosen start and end position, directions were queried, just as they would be by a driver. The directions were pre-processed to produce a trace common with the

[1] `http://maps.google.com`
[2] `http://www.osm.org`

framework. Resulting mobility patterns are seen to tend to use main roads rather than arterial roads.

Mobility patterns are thus more consistent with mobility patterns observed in reality, where a sizeable proportion of drivers use their GPS navigation systems to route themselves through a city. The prevalence of traffic along specific road sections is subsequently seen in results produced. The sum patterns of mobility traces for separate sample areas can be seen in Figure 4.8. The secondary, ETH trace dataset has been used by a selection of previous works [DMP+10, YLL+10]. ETH traces [NBG06a] represent realistic traces generated by the MMTS model. The MMTS model has been used to model the behaviour of the inhabitants of Switzerland using statistical census data. The traces represent 24 hours of vehicular mobility for a 250 x 250 kilometer coverage of Switzerland. Traces were enhanced from their raw form to form a high resolution dataset.

4.5. Collect-Merge-Share

To map the state of the road network, we develop the Collect-Merge-Share (CMS) protocol that first fragments available ATA *trace histories* and fits these fragmented travel times atop of a road network map. We assume that trace histories are aligned to available road maps, such that the traces more accurately follow mapped road sections. In effect noisy trace data is cleaned and filtered to produce more manageable maps (Figure 4.2). Trace histories provided imply direction using timestamps. We assume that the earlier timestamped data represent starting positions while ending positions are identified by more recent timestamps.

The acts of collecting, merging and sharing data are related to store-and-forward methodologies found in other protocols. The specifics of the travel time estimation aims require that we use bespoke components to transport and manage data. The framework architecture is modified slightly to provision the new application (Figure 4.3) - however the entire protocol relies on a limited number of instructions. In this form of the architecture the messaging feedback loop remains intact, while the actuation loop is removed. The protocol loop does not influence any actuators on board the vehicle and we do not allow the drivers to re-route themselves when provided new travel time maps. Travel times are transported using a modified Payload and managed using a MapStore component (Appendix B.1). The MapStore holds data-structures

Figure 4.2.: Real raw trace histories (left) and geographic map (right), illustrate the error inherent in present positioning methods. We assume that the vehicle is able to reconcile both datasets in pre-processing.

to index, merge and select travel time tuples for sharing. Sensor data and messages are collected at each computation cycle and presented to the loop which contains the Collect-Merge-Share (CMS) protocol. There are no modifications to the base framework architecture; rather additional analysis and application layers access the MapStore to visualise and graph travel time tuples. Each *Message* holds a single *Payload*, each Payload multiple travel time tuples. Travel Time tuples are expressed in the form: $[A, B, ATA, TS]$.

Here A and B waypoints represent the start and end positions of a road section (a travel time fragment of vehicle mobility). ATA represents the experienced and associated fragment travel time, TS the associated sample time time-stamp. For example, we represent a data filled tuple as:

$$[(51.50094, -0.1237), (51.50079, -0.11993), 32.2s, 235959 - 010111]$$

Read simply, the road section from 51.50094,-0.12370 to 51.50079,-0.11993 is measured to have taken 32.2 seconds to travel at 23h59 on 1^{st} January 2011. MapStore manages the *merging* and *selection* of tuples. Tuples are stored within the MapStore data structure and added to MapStore, either by a vehicle sampling or a vehicle receiving messages from neighbouring vehicles (sharing). In the first case, sampling produces new tuples. In the second case, sharing, messages are stripped of their Payload tuples and unique or not previously seen tuples are

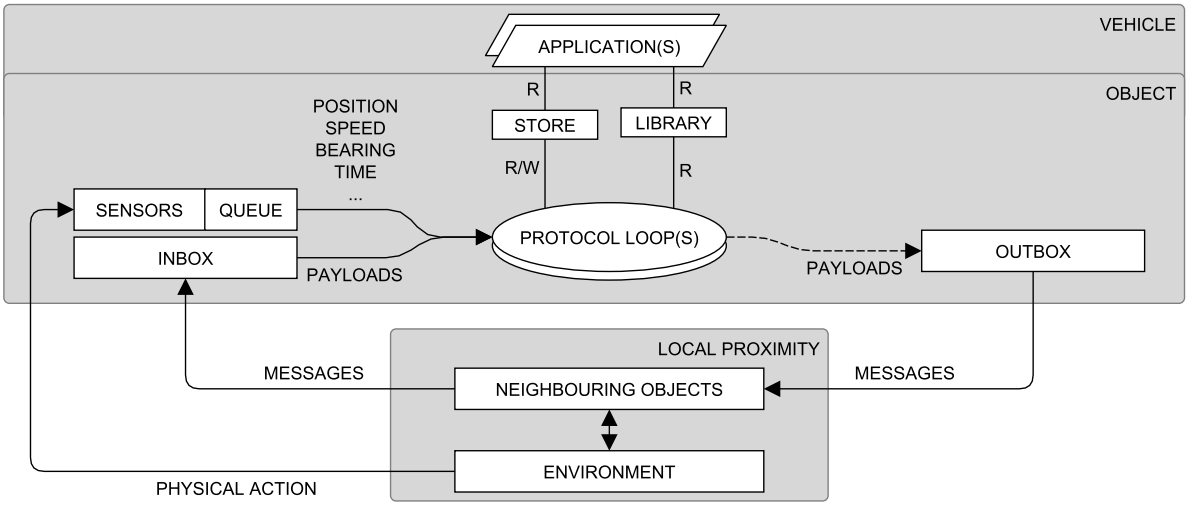


Figure 4.3.: Travel time estimation architecture. Shaded components are modified from the original framework architecture described in Chapter 3.

merged with the current MapStore travel time collection. Within this context, management is required to reduce the redundancy data stored. The protocol loop is provided read-write access to MapStore such that MapStore can be queried for subsets of elements. Within the architecture, a protocol can, on MapStore selection, construct new Messages for periodic broadcast. Layered with MapStore is the final application used to construct and visualise travel time graphs. While drivers are included in the architecture, our experiments measured and tested the application provision and not the final feedback actions of the driver. For instance, a driver would influence mobility by dynamically re-routing or changing the speed of the vehicle as the travel time estimation service operated.

Protocol execution is divided into three phases following the framework as a guide: (a) collection, (b) merging and (c) sharing (Algorithm 1). Our approach uses an epidemic opportunistic networking approach [VB00] and WAVE Short Messages (WSM). The Collect-Merge-Share (CMS) protocol is provided a large set of parameters including a set of received input messages (MI), a clock (t), a reference to the core Mobility object (R) and a *MapStore* data structure (S) for the storage and management of travel time tuple data. The clock object provides current time and stopwatch readings, where the stopwatch measures the elapsed time until it is reset. The stopwatch is required to measure the elapsed time taken to travel between waypoints.

During the collection phase (lines 1 to 6), a vehicle determines whether it has begun driving a new road section. To do this, the sub-routine R.`newSection()` matches the present position

Algorithm 1: Collect-Merge-Share

Input: A set of received messages MI, a mobility resource R, a MapStore data-store S, a clock t and a broadcast period bp

Output: A set of broadcast messages MO

```
1  begin
       // if beginning a new road section
2      if R.newSection() then
           // extract a road section fragment
3          A ← R.getWaypoint(previous)
4          B ← R.getWaypoint(present)
           // store travel time tuple
5          S.merge([A,B, t.stopwatch, t.now])
6          t.stopwatch ← 0
       // strip each message of its payload and merge each tuple held
       within the payload
7      for m ∈ MI do
8          for tuple ∈ m.payload do
9              S.merge(tuple)
       // periodically construct new broadcast messages using the
       store tuple selection
10     if isPeriod(t,bp) then
11         tuples ← S.selectTuples()
12         payloads ← tuples.split
           // allocate payloads to messages
13         for p ∈ payloads do
14             nm ← newMessage(R)
15             nm.payload ← p
16             MO.append(nm)
17     return MO
```

of a vehicle to the internal map and consults any previous positions stored within the mobility object (R). R maintains both records of mobility history and plans. Depending on the road topology, road sections tend to begin and end at road intersections. The vehicle calculates the distance between it and the route waypoints provided. If the vehicle has changed its road section, we construct a new travel time tuple using the previous-before-last and last route positions (A and B), the present elapsed-time stopwatch value, time-stamp and vehicle identifier. The tuple is *merged* with S and the stopwatch is reset.

As tuples are entered into the S they are compared to filter out cyclical data (redundant tuples) using a 'seen list' and insert road sections into the MapStore. We achieve this by using hashing tuples. For example, Figure 4.4 shows the merging process (steps 1 - 4) for an existing

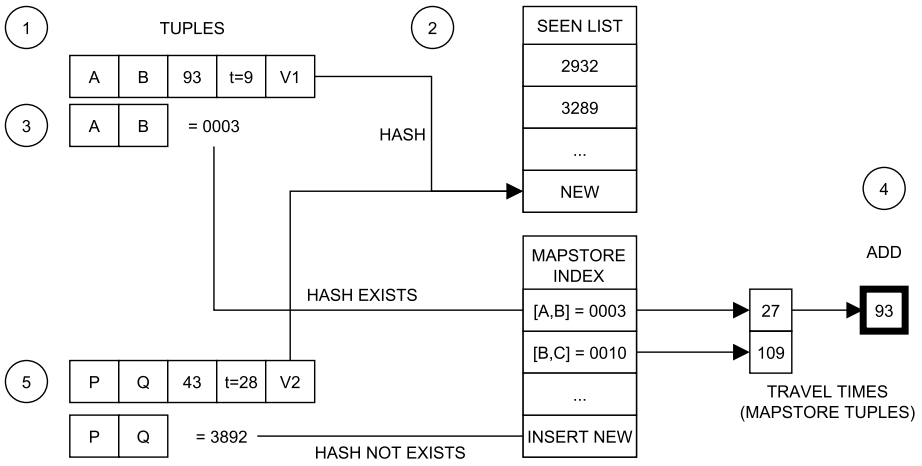


Figure 4.4.: Travel time tuple merging example. $[A,B]$ and $[P,Q]$ are first checked to determine if they are redundant. If not redundant samples, travel times are stripped and added to the MapStore.

road section ($[A,B]$) and a new road section ($[P,Q]$) in step 5. Travel times are associated with the particular road section. A separate index, not shown, stores the waypoint positions of AB and PQ. The merging phase (lines 7 to 9) is concerned with the addition of tuples from messages. Another hash is used to index and match road sections (for example $[A,B]$) to entries in the MapStore. Each road section in the MapStore contains a linked list of (multiple) travel time samples. In those cases where the tuple already exists but does not match the hash value, a new tuple is added to the MapStore (Figure 4.4).

As a single commuting vehicle is unlikely to spend all its time driving all roads (exceptions to this include buses, delivery vans and taxis), hence the sharing phase allows vehicles to collect travel times about other parts of the road network using ad-hoc message passing via broadcasting (lines 10 to 14). Sharing, itself, consists of three operations: (a) tuple selection, (b) message construction and (c) message broadcast. A selection strategy is used to determine which tuples to share with neighbouring vehicles for the next subsequent cycle. The method of selection has a bearing on which data is disseminated into the immediate vicinity. Hence selection methods affect the driving experience of a given vehicle. The selection of tuples is split into 1400 byte payloads and each new message is associated with a payload. Once messages are queued, they are ready for broadcast.

Selection strategies can be split into geographic and attribute-based strategies - thus selection criteria could be more complex. Figure 4.5 presents a geographic-based strategy, a radial

selection of tuples from MapStore. MapStore is queried to select stored tuples which exist within a certain distance of the present position of a vehicle. A variety of selection methods are possible.

Figure 4.5.: Example radial geographic selection from the point of view of single vehicle. We assume the vehicle to have already populated MapStore with tuples about the road network. The shaded region represents the travel times to be selected from MapStore for inclusion in a new Message.

In contrast, attributed selection could be made to query tuples which are recent or road sections which have at least M-many samples or filter specific samples made by a single vehicle (vehicle identifier). For the purposes of implementation, we use a radial geographic selection of the road network, thereby providing neighbouring vehicles with travel time tuples concerning the immediate travel space. Selection presents us with a means of *prioritising* the sharing of specific travel time tuples but at a trade-off of foregoing other data. A secondary limitation or filtration from selection is the construction of *Messages*. Thus, vehicles periodically broadcast a subset of their *MapStore* using the selection strategy employed - we could expect the strategy used within a system to be either homogeneous (all vehicles use a single strategy) or heterogeneous (many different selection strategies). Data is disseminated according to a store-and-forward paradigm as commonly found in opportunistic networks.

It is notable that each vehicle holds its own subset of the known road network data as a result of the roads driven and the interactions between vehicles sharing road network data. For example, considering a set of vehicle MapStores S_{V1}, S_{V2}, ..., S_{VN}, where N represents the number of vehicles within the system, it is unlikely that the MapStore of one vehicle will be the same as the MapStore of another ($S_{V1} \neq S_{V2}$).

4.6. Estimation and Mapping

While a protocol is responsible for the collection and sharing of travel time tuples, the analysis of travel time estimation is applied within the application layer. The application processes data by first estimating the travel times and then associating travel times with road sections. This section discusses the estimation algorithm and the maps produced by the service.

4.6.1. Estimation Algorithm

While the CMS protocol is operating, we can assume that the estimation service residing on each vehicle is provided with a road map, the local MapStore and common libraries which contain methods to estimate travel times. As the MapStore contains road sections and associated travel time lists, the estimation of a route's estimated travel time is a combination of both *known* and *unknown* travel times. When provided a route to estimate, this is broken into a set of mobility fragments. These fragments are matched against sampled travel times (stored tuples within MapStore). If one or more sampled travel times can be associated with a fragment entry - the pair is said to be *known*.

If more than a single sampled travel time fragment exists for a road section, we interpret the travel time as an arithmetic mean, the sum of sampled travel times for a particular road section divided by the number of samples. Included in the calculations are minimum and maximum travel times. Where no sampled travel times exist for a road section, we denote the road section as being *unknown*. In unknown cases, we revert to a predictive calculation - estimates are calculated as the optimal travel time to travel the particular road section (with knowledge of the speed limits). The approach attributes costs to the usage of a road section and a number of extended strategies could be used to estimate the travel time for a given road section, including linear or exponential weighted moving averages. The accuracy of a mean is improved by providing a measure of deviation using maximum (worst case) and minimum (best case) travel times. It is unclear which estimation of travel time is best suited for the application. To find a best method may require that we analyse the usage of strategies and their outcomes.

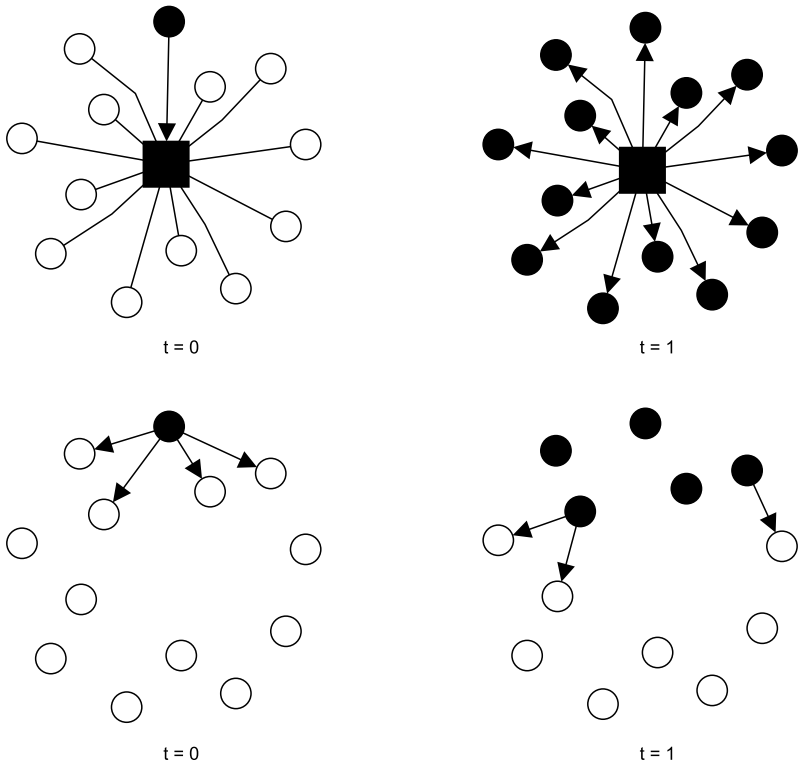
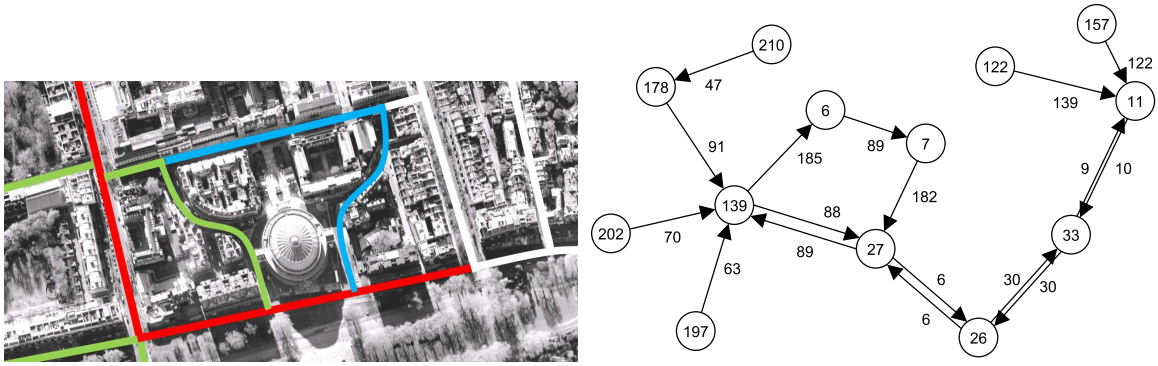


Figure 4.6.: Data availability problem. A centralised approach to data sharing with a central authority (top), and a decentralised approach (bottom). Filled nodes represent data which has been shared between a source in successive time-steps from 0 to 1.

4.6.2. Data Recency and Availability

A primary goal of the framework is that it produces a working decentralised vehicular service via the stipulation of a protocol. A set of measurements is used to analyse the protocol performance in relation to centralised services. We divide the evaluation of the decentralised service into (a) MapStore, (b) messaging and (c) visualisation issues. Decentralisation highlights a necessity to understand both the behaviour of the system as a whole and the behaviour of individual vehicles, which through local behaviour generates a global behaviour. Behaviour in this context refers to the actions or refinements of the system under specified scenarios. For example, how do the actions of vehicles change the sharing of data? What parameters improve performance? What are the trade-offs of altering parameters? How does the decentralised service compare to the centralised service?

Central to the performance of a travel time estimation service is measuring the *recency* and *availability* of data. While both centralised and decentralised vehicular services seek the same



(a) Road network. (b) Travel times map overlay.

Figure 4.7.: Travel Time map visualisations.

goal, the overheads associated with decentralisation are not immediately obvious. For example, naive simple services may upload their data to a central authority in a single transmission, yet they also require a download of shared data in a successive time-step. If we assume centralised services query while simultaneously updating, we consider this action to cost two transmissions. Once data has been shared within a decentralised setting, it is available locally to a vehicle. This requires only a single broadcast. The trade-off is that decentralised data then resides in distribution and is unavailable to neighbouring vehicles which may seek to use such data in the next time-step. Hence data in decentralised systems is inherently stale. This data availability problem comparison is visualised in Figure 4.6. Our experiments seek to determine the speed of sharing collected ATA data. Depending on the user of a map, the mapping algorithm used may be different. For instance, maps provided to a driver are generally simpler in form to those provided to city administrators, traffic authorities and city planners. The solution presented uses provided maps to highlight positions of interest. Figure 4.7(a) illustrates a road section within the mean travel time graph depicting travel times as edge weights - the graph removes the spatial association with the map. Such graphs can be presented to a routing algorithm to find the shortest route of travel between two geographic positions. The graph can be matched to the road network and coloured to represent the flow or lack of flow existing along a road section (Figure 4.7(b)). Similarly, mean travel times can be replaced with other MapStore data, such as tuple recency, vehicle samples, sample sizes (the number of samples for a particular road section), minimum, maximum and unknown/known travel times.

4.7. Evaluation

Within the following section we show and evaluate a subset of results specific to exploring scalable Collect-Merge-Share (CMS) performance. To test the feasibility of the CMS protocol, the protocol was prototyped and simulated using our framework's Geographic Urban Simulator (GUS). We highlight a subset of scenarios which include (a) a synthesised *grid*, (b) two synthesised mobility datasets constructed using OSM maps (for the cities of London and San Francisco) and (c) a portion of mobilities from the ETH trace dataset [NBG06b]. While the results produced were exhaustive, for the four specific mobility scenarios, we focus on the performance of the protocol in terms of adapting mappings and the message feedback cycles produced. Reiterating the problem, the aim of the service is to collect, share and map the current or near current state of the road network, given available data. The intended service result is the production of travel time maps and graphs which represent the state of the road network. Maps can be used or referenced visually by the driver of a vehicle and graphs may be interpreted by a navigational computer for routing. These maps and directed graphs can thereby be used by various off the shelf routing algorithms to re-route a vehicle as travel time tuples are shared [Dij59, SSV08]. We do not consider the outcomes of re-routing during travel and the consequence of feedback issues arising. Rather the travel time application presents itself as an example system to which the framework can be applied and measured. Due to the dynamic mobility and change in mobility of vehicles, *adaptive* and *multipath* routing approaches present themselves as possible solutions to dynamic routing problems.

4.7.1. Simulation Parameters

Simulation in our framework requires the provision of *mobility trace patterns*, *simulator parameters*, *protocol parameters* and a *prototype protocol* (decentralised service). The mobility patterns used vehicle mobilities within a 2.5 by 2.5 kilometer geographic region (Figure 4.8)[3]. Synthesised mobility patterns were computed using scraped Open Street Map (OSM) and Google Maps data (Appendix A.2). Vehicle routes used either simple or extended versions of Dijkstra's algorithm for single-source shortest path traces [Dij59, DSSW09]. The mean Estimated Time of Arrival expected for vehicles travelling at an average speed of 8 meters per

[3]We chose to use this confined region and limited area for simulation as these limits had been specified in related works.

second (28.8 kilometers per hour) was specific to the mobility pattern used. Vehicles either travelled through the region or drove to specific positions within the region.

Table 4.1 considers the statistical properties of grid and city mobilities used, where each mobility set specified more than 1000 patterns. The table shows the maximum, minimum and mean travel times for each road network sample set.

Region	Minimum	Maximum	Mean	Std. Deviation	Mean Travel Distance
Grid	25s	742s	267s	151s	2.1km
London	17s	575s	199s	100s	1.5km
San Francisco	33s	545s	166s	85s	1.38km
Zurich	18s	509s	190s	97s	1.52km

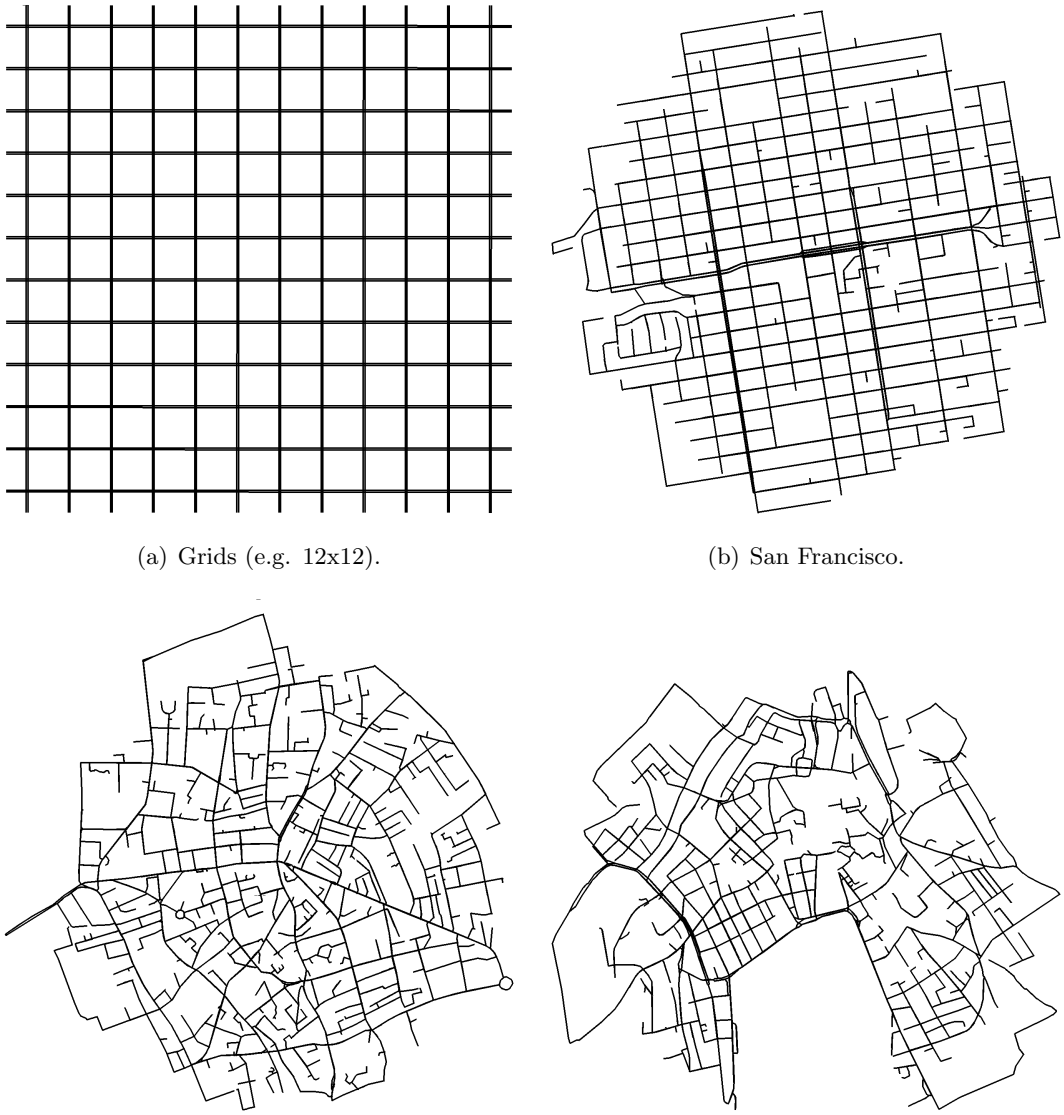
Table 4.1.: Input mobility patterns: each regional dataset used contained over 1000 vehicle movements. Values represent the 'ideal' mobility travel times.

Protocol parameter settings were homogeneous to all vehicles within the road network. The protocol processing cycle was repeated every 0.1 seconds (100 milliseconds) with broadcast windows[4] specified for 12.5, 25 and 50 second time-outs. All four example road networks differ visibly in their structure. The San Francisco sample area was most similar to synthesised grids containing many regular square blocks, while London and Zurich represent organically constructed road networks - road networks which grew over time without significant planning by city planners.

WAVE has a theoretical maximum communication range of 1000 meters. Previous performance work by Kai-Yun et al. [HKHL10] suggest that the efficient communication range for WSM exists within the 100 to 300 meter range, where a 3 Mbit/s throughput yielded a maximum packet loss of 9% [IEE10, Eic07]. Given these throughput and packet size values we calculated a maximum broadcast or upper limit of 30 Payloads per second for travel time data. The broadcast interval was randomised to avoid transmission collisions with other communicating vehicles. Message loss percentages were applied to complete messages (i.e. we drop complete WSMs instead of packets, but follow previous works).

Simulation parameters specified a constant population of vehicles within the road network at any time. In other words, as vehicles exited the road network, new vehicles are created and introduced into the road network to maintain the vehicle population within a region. The sys-

[4]Windows represent time slots within which a vehicle may broadcast messages.



(a) Grids (e.g. 12x12).

(b) San Francisco.

(c) London.

(d) Zurich [NBG06b].

Figure 4.8.: Road networks (2.5 x 2.5km).

tem was expected to adapt to these vehicle losses and increases. Vehicles were set a maximum speed of 8 meters per second (28.8 kilometers per hour following United States Transportation guidelines [Mas06]). Vehicle movement was restricted to the waypoints, direction and speed limit provided when a vehicle began its journey. Travel times stored with a vehicle's MapStore were provided "unlimited" storage capacity to reflect the likely storage capacity that future vehicles might have (in the order of several hundred gigabytes). Each travel time tuple was stored in a frame format (Table 4.9). Given a total tuple size of 44 bytes, a vehicle can potentially broadcast 11700 tuples per second. We used a maximum broadcast period of 25

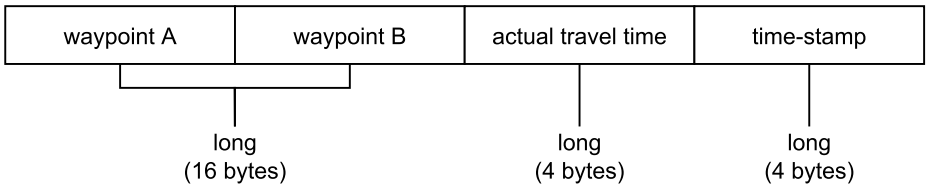


Figure 4.9.: Travel Time payload.

Parameter	Symbol	Value	Unit
Communication Range	CR	200	meters
Maximum Random Broadcast Interval	BP	12.5, 25, 50	seconds
Maximum Speed	S	8	meters per second
Tuple-selection radius (random)	TS	2500	meters
Received Message Loss	RML	0 - 50%	messages

seconds and considered 5, 15 and 25 second broadcast period, chosen as these vehicle speeds corresponded with the potential opportunity for vehicles to make contact given their relative speeds and broadcast range[5]. A lower maximum broadcast interval was expected to increase the probability of communication between vehicles. Each experiment was repeated ten times for each scenario. The experiments presented consider varying parameters and hence mean results.

By monitoring the MapStore we are indirectly able to interpret interactions between vehicles, the types of roads vehicles are driving (straights, corners or high density blocks) and predict when vehicles are likely to retrieve updated estimations. In the remainder of this section we highlight the (a) message numbers, (b) MapStore growth, (c) route discovery times and (d) visualise the travel time estimation maps as heatmaps.

4.7.2. Message Counts

Table 4.2 shows the typical mean broadcast message counts per minute for increasing vehicle populations versus increasing broadcast period (BP) for various cities. Vehicle populations (POP) increased between 250 and 1000 vehicles per 2.5 x 2.5 kilometer area. Notably, fewer messages are broadcast than the ideal, as some vehicle mobilities are shorter than others, where mobility patterns are influenced by the city road network.

[5]We used statically chosen broadcast periods based on the number of times a vehicle might make contact with a communication range of 200 meters at a speed of 8 meters per seconds. Broadcast period could be optimised

Map	BP	Number of Vehicles (POP)			
		250	500	750	1000
Grid	12.5s	1173	2320	3540	4732
	25s	477	1121	1702	2282
	50s	260	500	842	1174
London	12.5s	1159	2378	3567	4782
	25s	554	1150	1761	2350
	50s	276	557	865	1158
San Francisco	12.5s	1176	2351	3245	4670
	25s	582	1166	1751	2372
	50s	286	568	852	1179
Zurich	12.5s	1152	2378	3565	4754
	25s	567	1152	1757	2383
	50s	253	567	881	1170

Table 4.2.: Mean broadcast message counts (per minute) for increasing vehicle populations (POP) versus increasing broadcast period (BP).

Table 4.3 shows the ratio of received messages to broadcasts for varying vehicle populations (POP) and varying BP. The value represents the redundancy or number of copies made, per broadcast. For example, for the city of London, given a 12.5 second BP, payload data was copied a mean of 14 times for each broadcast made given a population of 250 vehicles. Varying the BP did not significantly effect the receive-broadcast ratio (R:B). However, for each city road network a BP of 25 seconds was seen to raise the R:B factor slightly. For example varying BP for Zurich yielded a 4.4% increase in retrieval for a vehicular population of 250 vehicles. Notably we see the reverse effect in larger vehicle populations. While we use a set or unchanging BP, the result suggests that the BP can be optimised or varied to maximise the R:B for a given road network or city domain. For increasing vehicular populations we saw a linear increase in the R:B ratio. Hence a direct relationship exist between increasing vehicle populations and message dissemination. Improving dissemination reduces the staleness of travel time data.

Figure 4.10 shows the mean number of messages received per minute per vehicle for each road network, where we vary the broadcast period and increase the vehicular population within the road network. All road networks show a declining R for increasing broadcast period; however, within the cities of London and Zurich, message retrieval finds a steady performance for populations of 1000 vehicles (marginal R increase approaches 0). Hence, performance is changed little for increasing period. Using this information, the service can use this information

to change as vehicle speed changes.

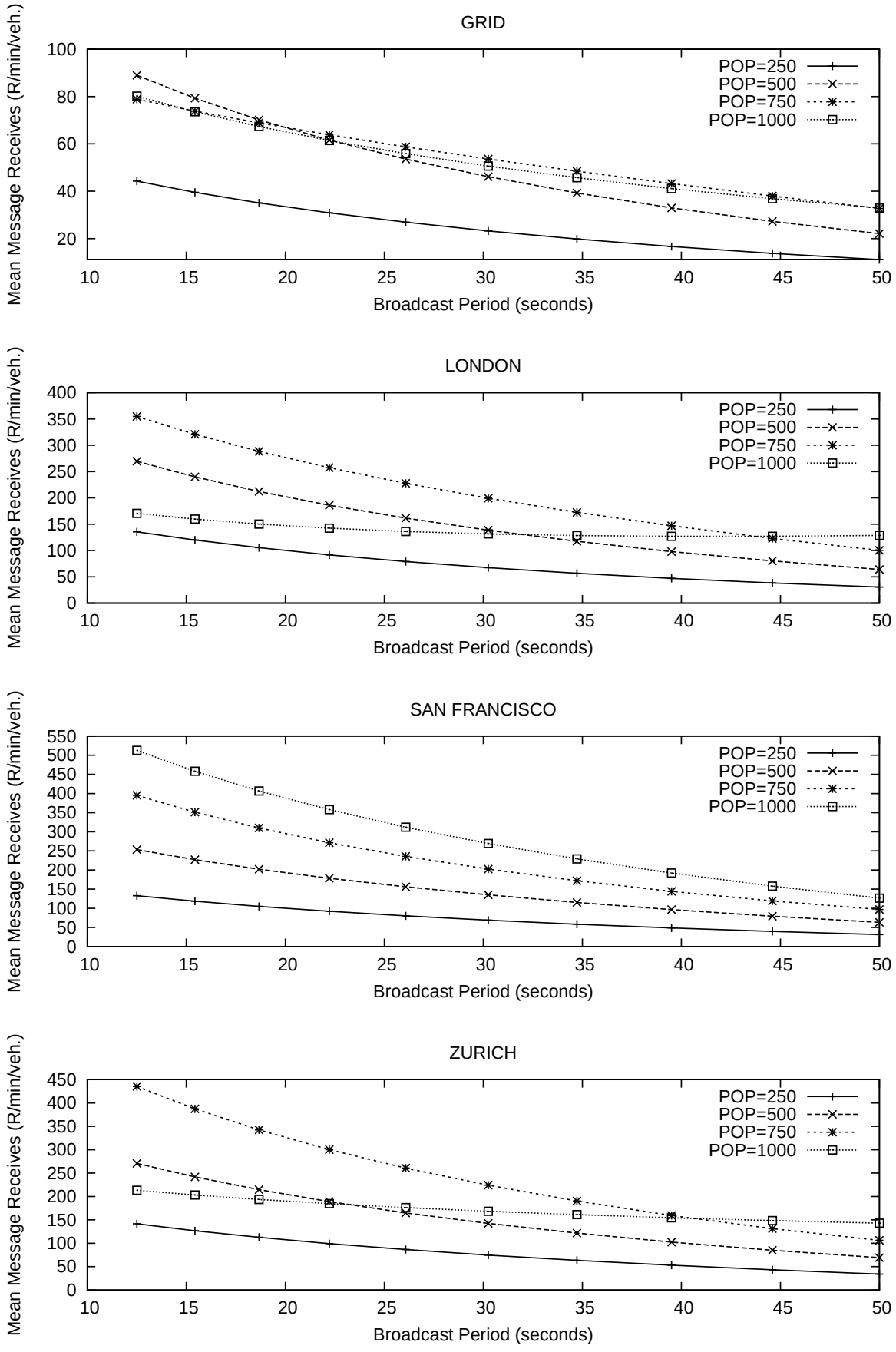


Figure 4.10.: Comparison of mean message per vehicle per minute retrieval for different road network topologies for increasing broadcast periods (seconds).

Map	BP	Number of Vehicles (POP)			
		250	500	750	1000
Grid	12.5s	4.69	9.45	14.32	19.51
	25s	4.79	9.41	14.13	19.40
	50s	4.79	9.63	14.23	19.21
London	12.5s	14.38	28.61	42.45	62.32
	25s	13.61	28.5	42.61	61.73
	50s	13.29	27.94	44.12	56.55
San Francisco	12.5s	14.1	27.05	42.04	54.50
	25s	14.3	28.47	41.47	56.30
	50s	13.83	28.13	42.99	55.83
Zurich	12.5s	15.11	28.73	46.34	61.84
	25s	15.8	29.36	45.56	61.21
	50s	15.12	30.51	46.87	61.62

Table 4.3.: Mean ratios of received messages to broadcasts (R:B), given Table 4.2, for increasing vehicle populations (POP) versus increasing broadcast period (BP) for varying road maps. Each value represents the mean redundancy for each message broadcast.

to reduce message redundancy and messaging overheads, as the increased population serves to balance the need for increased messaging.

A result of scaling or increasing vehicle population is that the probability of message collisions increases. To avoid message collisions the CMS determines a broadcast time from the broadcast window. Effectively the broadcast time is calculated at a time less than or equal to the broadcast window size. Depending on the number of vehicles receiving a message - a broadcast copies tuples to other vehicles. As the population of vehicles within the cities is increased we see a linear increase in the multiple of messages received versus broadcasts made. Effectively, increasing the population or density of vehicles in the city increases the number of connections between vehicles and thereby improves the rate at which data can travel through the vehicular network. The trade-off cost of this increase in vehicle population is increased pressure on limited bandwidth.

In summary, the R:B ratio growth is linear for all road networks. However, the marginal increase in R:B multiple is steeper for city road networks than grids. This is largely due to the constraints of mobility due to realistic road networks. Real road networks limit the number of alternative routes which a vehicle might use and thereby improve the facilitation of message dissemination.

4.7.3. MapStore Growth

In the previous subsection we showed the effect of (a) varying BP and (b) varying vehicle populations on the retrieval performance of vehicles. MapStore growth is in turn reliant on the retrieval of tuple payloads from neighbouring vehicles. An understanding of the tuple inflows into MapStore presents a means of understanding the outcomes associated with specific protocol design choices. MapStore growth is monitored by considering the MapStore *index* and *unique* travel time fragment counts. MapStore has two means of growing, namely through (a) the sampling of tuples or (b) the sharing of tuples. The MapStore held by a vehicle represents the known state of the road network in a previous time instant from the view point of a single vehicle. As tuples are shared the state of each vehicles MapStore is normalised towards an average MapStore state. Figure 4.11 compares index (i) and unique fragment growth (u) for the four cities. The figures represent a global comparison of MapStores for vehicles driving a city road network.

As multiple vehicles explore the road network the entire road network is both labelled and ATA associated. MapStore index is seen to grow towards a maximum count, indicating that vehicles are successfully associated with at least a single tuple with each known road section. In contrast, the count of unique tuples continued to grow linearly, as updates are made about the state of the road network. In this context, the MapStore index and the MapStore fragment counts are separate.

Each index represents the number of road sections known by a vehicle. Each unique tuple made unique as it represents a road section, but a sample at a different timestamp (a one-to-many relationship). Expressed as a hash-map, the index represents the keys of the MapStore while the unique tuples represent the actual tuple values stored. The index of each map has an upper bound, specified by the number of unique identifiers associated with road sections.

Hence, the size of the MapStore index shows us when the community of vehicles has successfully found at least one travel time for a discovered road section. Notably the index is rapidly found, providing a gauge on how long it might take a new vehicle to learn about the travel times associated with a given city area. The more intricate the road map, the more road sections and therefore the longer the time required to map the entire city area. We see this in comparing the maximum index (max-i) curves for the grid and city maps. For instance, San

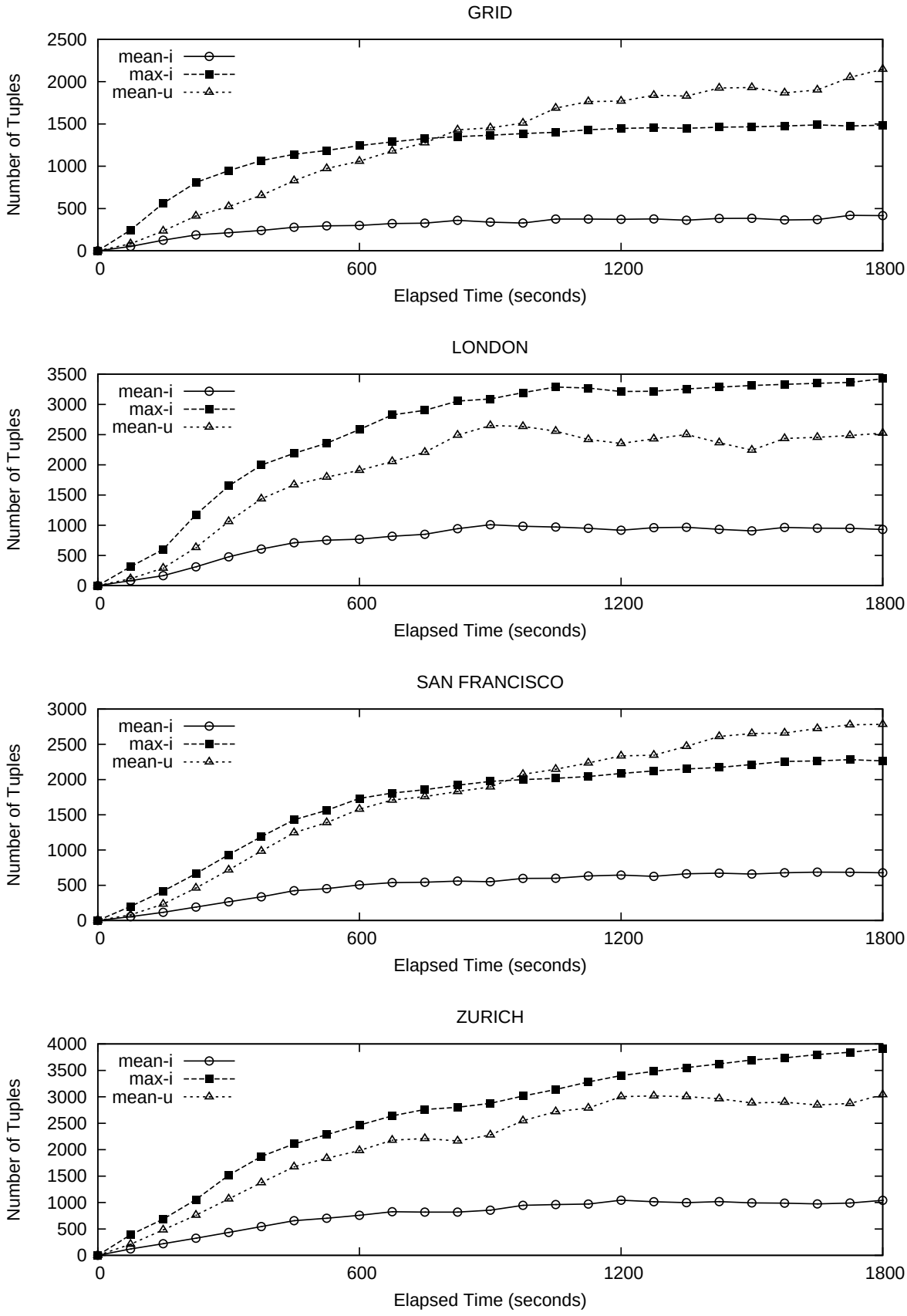


Figure 4.11.: MapStore growth (POP=250, CR=200m, BP=12.5s, RML=0%) for 1800 seconds of simulation time.

Francisco (a less intricate road network) peaks at approximately 1000 tuples, while London peaks at more than 2000 tuples. In each plot we also compare the mean index. The mean is significantly lower, due partly to the fact that new vehicles enter the road network without any knowledge or mapping of the local space. The mean number of unique tuples (mean-u) is seen to grow linearly but degrade similarly as time progresses. Mean-u distorts at a longer time interval as a consequence of vehicles exiting the road network. As vehicles exit tuples which have not been shared cease being available to the community of vehicles. MapStore growth results point to vehicles rapidly mapping their space. These new vehicles skew the plotted MapStore. However, from these plots we can ascertain that individual MapStore sizes exist between 0 and max-i values.

The growth of MapStore is misleading in terms of its functional growth. While the curves exhibit the properties of logarithmic growth (of the form $y = \ln(x)$), individual vehicle stores grow and fit fragments as successive averaged logistic curves of the simple form:

$$M(t) = \frac{1}{1 + e^{-t}}$$

The tuple size ($M(t)$) is functionally associated with time (t). The logistic curve allows us to predict the performance of the MapStore by monitoring the derivative growth of both index and unique tuple values. Given the sample road networks it took less than 837 seconds (approximately 15 minutes) for vehicles in each road network to learn more than 80% of each road network. While MapStore growth is seen to reach a maximum level once all road sections have been indexed - it is notable that logistic growth occurred for subsequent time steps. Effectively MapStore growth is continuously occurring as old road sections are re-driven and vehicles update their data about a particular region. The logistic function produced suggested that when the mean MapStore growth rate becomes zero the vehicle holds data about all road sections within the sample region. Hence, similar behaviour and MapStore growth is repeated as time progresses and the new state of the road network is sampled.

A trade-off of decentralisation is that once tuples are collected they remain distributed until shared. This effects the accessibility of tuple data and hinders the travel time estimations to consider old data. In contrast, centralised approaches instantly upload collected tuples in a central and commonly accessible central authority. Data used in centralised estimation would

therefore be less stale than decentralised data. Previously, MapStore index results suggest the minimum required time to discover at least one tuple for each road section, represented as the time interval where the derivative MapStore index growth, tends towards 0 or flattens (Figure 4.11). This measure seeks to find where the community of vehicles give each other at least one ATA for each road section within the road network.

4.7.4. Route Discovery

Following from known and unknown travel estimation, we measure the distributions of times required to estimate travel times using only ATAs after 1800 seconds of service operation. In other words, how long does it take for us to make estimations using only known travel times? The Route Discovery Time (RDT) represents the elapsed time taken to completely estimate a route from *known* sampled travel times held within the MapStore. RDT is highly dependent on the road network, a vehicle's route, the number of contacts occurring between a vehicle and its neighbours and the speed of a vehicle.

Road network topology influenced RDT as vehicles were bottlenecked to follow specific mobilities. The grid structure presents the most divergent road network and as such RDT performance was worst in the network. The mean RDT requiring 216.4 seconds with a standard deviation (SD) of 121.8s. The cities of London and Zurich had similar RDT performance. Mean RDT requiring 139.8 seconds (SD = 92.7 seconds) in London and 189.4 seconds (SD = 96.4 seconds) in Zurich. San Francisco performed best with a mean RDT of 78.4 seconds (SD = 71.4s).

As MapStore growth showed, vehicles initially entering a new road network were required to sample the road network. Hence the time taken to perform route discovery is longer, because few samples existed in the early operation of the service. As MapStore grows to its maximum achievable level, the subsequent RDT is reduced. Hence, in an unexplored road network RDT is high, while in an explored network RDT is short (less than the maximum previous RDT), however data may be stale. This short RDT is shown by the resultant frequency of RDT expressed as a percentage of discovery times (Figure 4.12).

For example, for the city of London, 9% of travel time estimations could be based on ATA data, as vehicles entering the road network are instantly provided at time 0. Another 25% of vehicles found estimations relevant to them within 60 seconds. In comparison, more than 60%

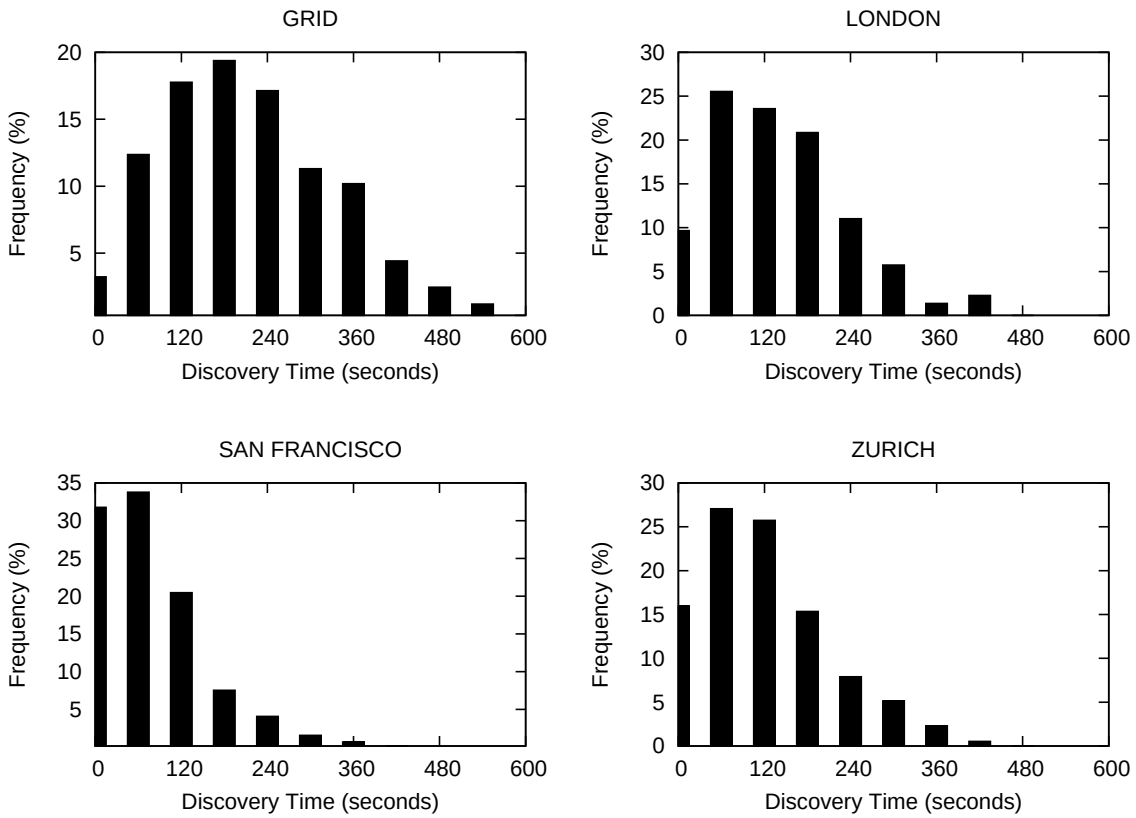


Figure 4.12.: Route Discovery Time (RDT) distributions for selected city mobilities for 1800 seconds of simulation (POP=250, CR=200m, BP=12.5s and RML=0%).

of vehicles within the San Francisco region discovered relevant ATAs relevant to their journeys within 60 seconds. For each mobility, each differing road network, the outcome was similar. As time progressed the normal distribution of RDT shifted from right to left (i.e. closer to shorter discovery times). We see a large number of routes (purely estimated on ATAs) found within 0 to 60 seconds of a vehicle entering the road network. More extended RDT was seen where vehicles were travelling longer routes (hence the RDT was typically longer) and where vehicles were initial explorers of the road network.

The distributions specifically show the times required for a vehicle to find estimations relevant to its own journeys. For the cities, these times are significantly rapid, with vehicles finding ATA route estimations within less than 5 minutes. RDT performance is seen to be skewed in favour of cities as city road networks typically constrain more vehicles to use main roads as they travel through a region. Grid RDTs, while shifting left, take longer to find, as the grid road networks are less constrained compared to the city road networks (vehicles fan out more

evenly).

4.7.5. Maps for Travel Time Estimates

The final objective of the service is to map travel time estimates. We provide two mappings, namely heatmaps and graphs. Historically, mapping has provided an accessible means by which to view summarised data. As each MapStore is unique (due to previous contacts, events and dependencies) each vehicle map represents its own view of the state of the road network at a particular time. In comparison to a centralised service, all data is uniform between vehicles, as all vehicles use a single data source. The maps are an approximation of the true state of the total fragments available. Figures 4.13, 4.14, 4.15 and 4.16 visualise traffic overlays for a subset of vehicles. The maps represent the known state of the road network after 15 minutes of operation. Specifically the maps shown represent the number of fragments known for a particular road section (i.e. the number of tuples and values associated with a particular road section index). Heatmaps are an accessible means by which drivers and city administrators may interpret the state of the road network [CRK08]. The hotter a region of the road network, the more a vehicle knows about that road section. Interpreted in an alternative way, heatmaps present the density of vehicles within varying locations of the road networks considered. A driver may seek to avoid well known (sampled) road sections and choose routes using less well known routes or vice versa based on choice of data. A decision on how to interpret the data is dependent on the driver. The same maps might be interpreted by city administrators to route or reorganise traffic flows using road rules. Other data could similarly be mapped and presented in the same manner.

In terms of performance, heatmaps highlight regions of road which form traffic bottlenecks. For example, the commonly modelled grid is seen to linearly distribute traffic (Figure 4.13). The absence of bottlenecks in the grid means that there may be little benefit in strategically routing along the grid given the dispersal of traffic. As vehicles are distributed linearly due to the regularity of the pattern, R:B was a multiple lower than that found in city road networks. The road networks of cities such as London, San Francisco and Zurich main roads represent bottlenecks in cities which benefits service provision as more vehicles are available to share data along a particular arterial road. While London and Zurich demonstrate singular road bottlenecks (Figures 4.14 and 4.16), road networks within the grid and San Francisco both distribute

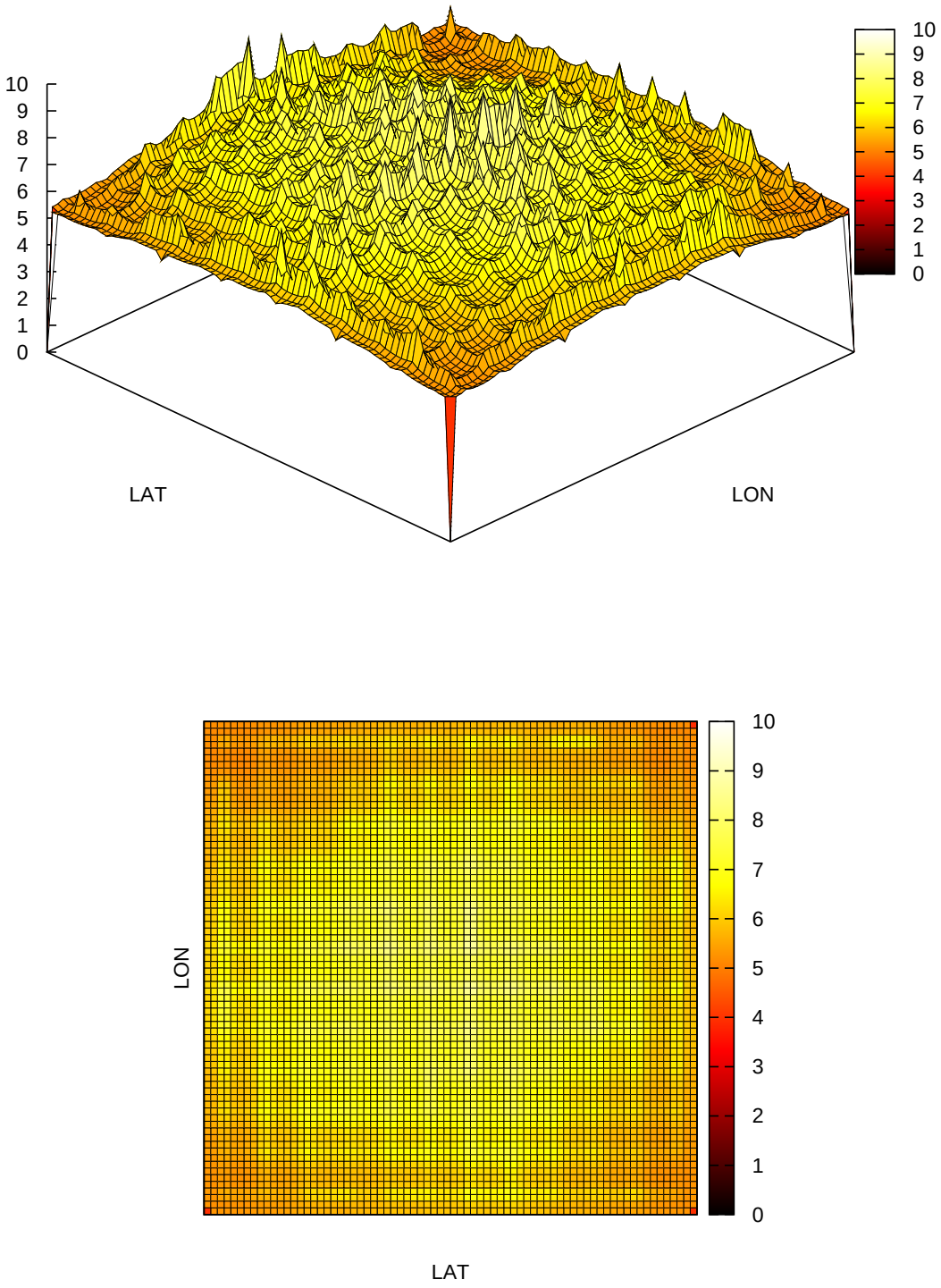


Figure 4.13.: Grid heatmaps showing surface (above) and mapping (below).

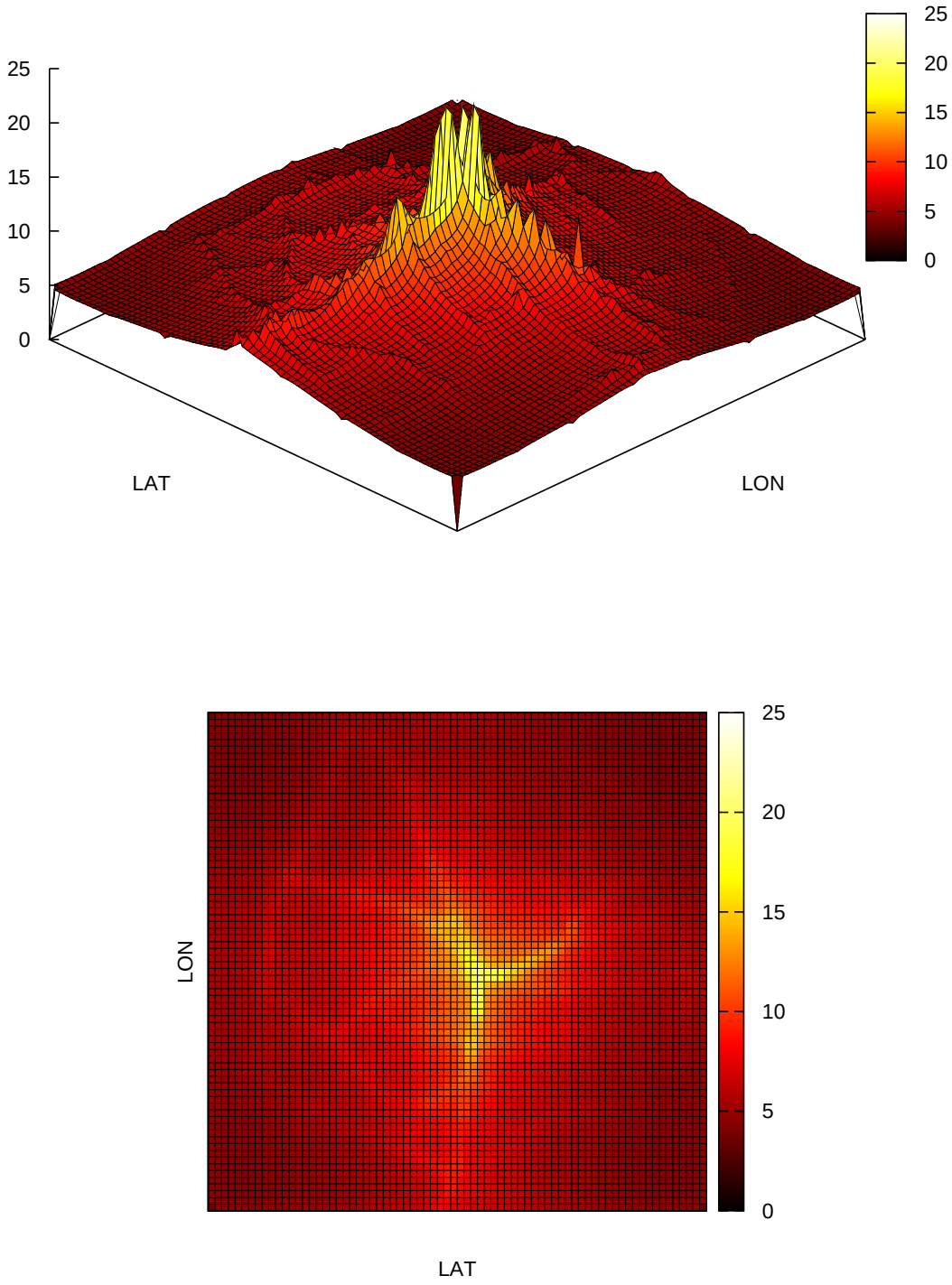


Figure 4.14.: London heatmaps showing surface (above) and mapping (below).

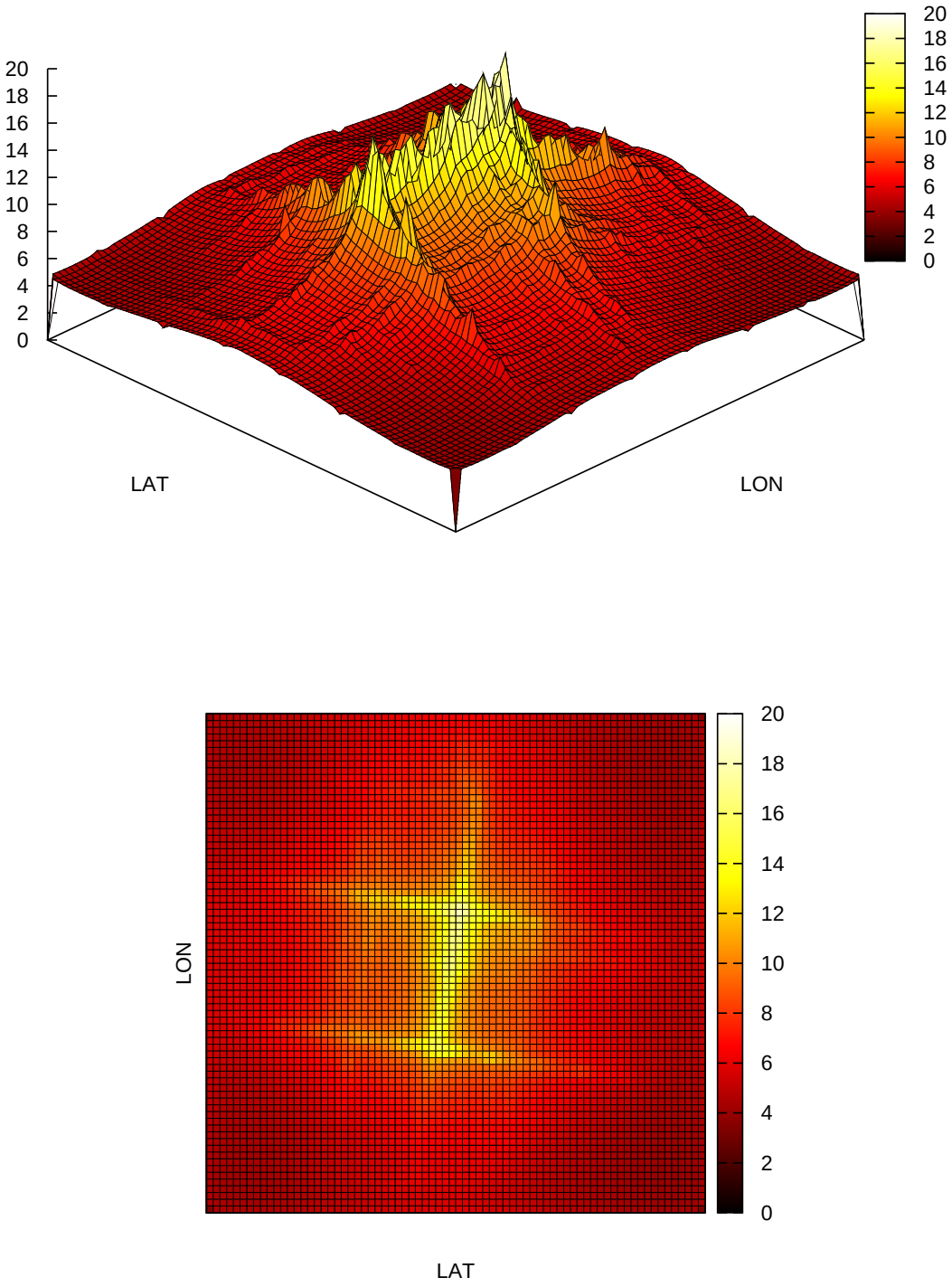


Figure 4.15.: San Francisco heatmaps showing surface (above) and mapping (below).

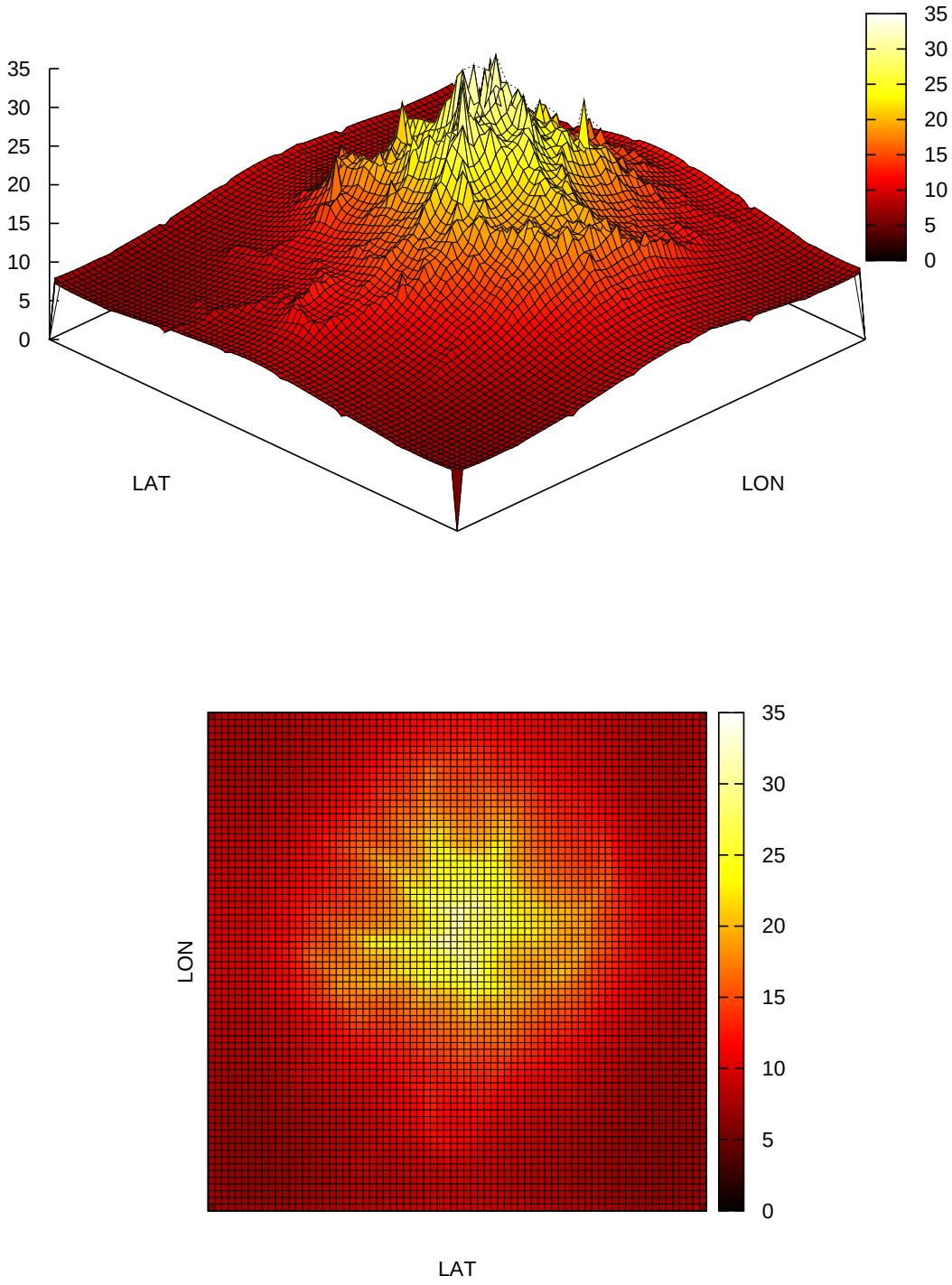


Figure 4.16.: Zurich heatmaps showing surface (above) and mapping (below).

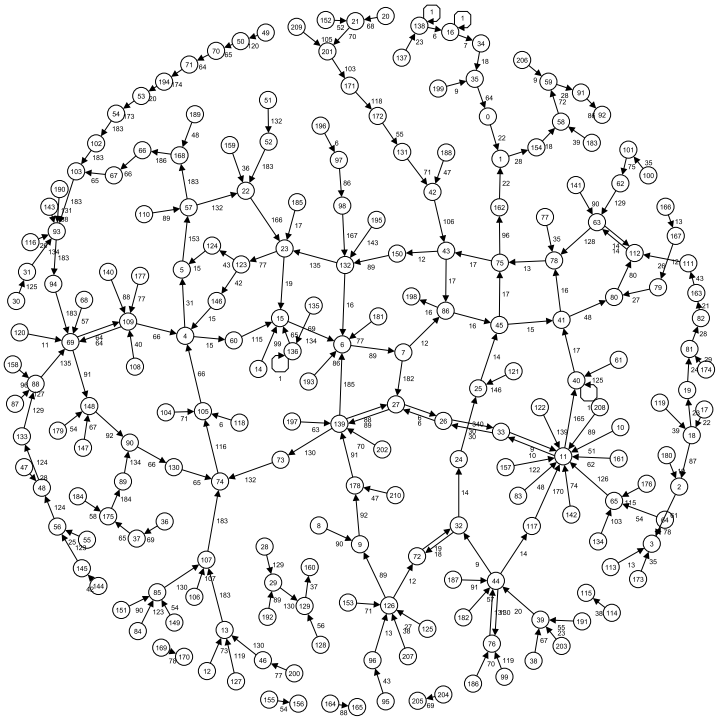
traffic to secondary intersections (Figures 4.13 and 4.15). The occurrence of bottlenecks allows re-routing methods to more adequately identify those road sections to be avoided.

The heatmaps generated are produced from the same data as the networks illustrated in Figure 4.17. Figure 4.17 shows the state of the MapStores of two vehicles (X and Y) at times varying times (143.9 and 561.8 seconds) for the city of San Francisco. We use directional graphs to store the road network state. In comparison, the two graphs differ in terms of the number of known road sections (graph X is smaller in size compared to graph Y as fewer tuples have been sampled and shared). As time progresses, the MapStore grows and the visualised road network increases in size. These road network graphs can be used without modification for the dynamic re-routing of a vehicle using off-the-shelf routing algorithms.

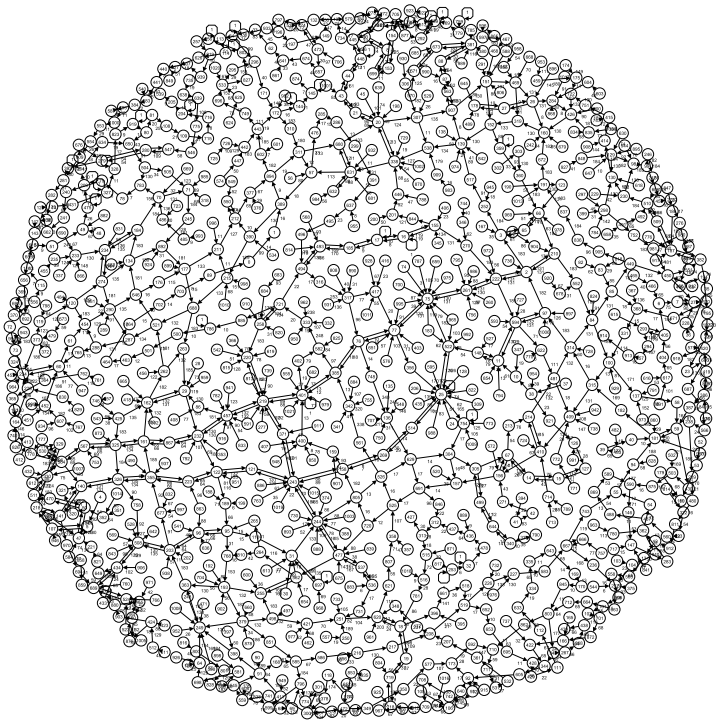
4.7.6. Message Failure and Redundancy

To determine the effect of dropped messages on the service, we simulated received link failures as received message losses (RML) of between 0% and 50% of messages retrieved (R). Figures 4.18, 4.19 and 4.20 select a subset of dropped packet experiments and considers the effect of message loss on the growth of the sizes of MapStore index (mI) and unique tuple counts (mU), namely ($|\text{mI}| \leq |\text{mU}|$). In Figure 4.18 the MapStore index is affected only slightly by increased RML. However in Figure 4.19 we see that the number of unique travel times stored inside MapStore declines significantly beyond at 25% RML. A smaller mU considers that while MapStore may map a regional road map, the richness of data is limited. For example, for the city of Zurich, mU declines from 2254.69 fragments for 0% message loss to 1748.58 fragments for 50% loss. Hence a typical vehicle has fewer ATA to compare to one another and therefore the quality of 'known' estimations are lowered. As expected, this mU decline is partly due to fewer redundant tuples surviving successive hops between vehicles (Figure 4.20 shows the tuple redundancy count - the count of the number of tuple copies which are discarded during operation). The decline of mR is most significant in Zurich as the road network is more constrained than other road networks from the set. For increasing RML, RDT was seen to shift right, thereby highlighting reduced performance.

We see an inverse relationship between MRL and redundant fragment counts (mR). As increased numbers of messages are lost, the number of redundant fragments available to vehicles declines. The significant redundancy in the system is negative such that extra bandwidth,



(a) Vehicle X after 143.9 seconds of travel holds 232 fragments.



(b) Vehicle Y after 561.8 seconds of travel holds 1290 fragments.

Figure 4.17.: San Fransisco Travel Time road network graphs. Edge labels represent the most recent travel times for an associated road point. Each node represents a labelled geographic position (POP=250, CR=200m, BP=12.5s, RML=0%).

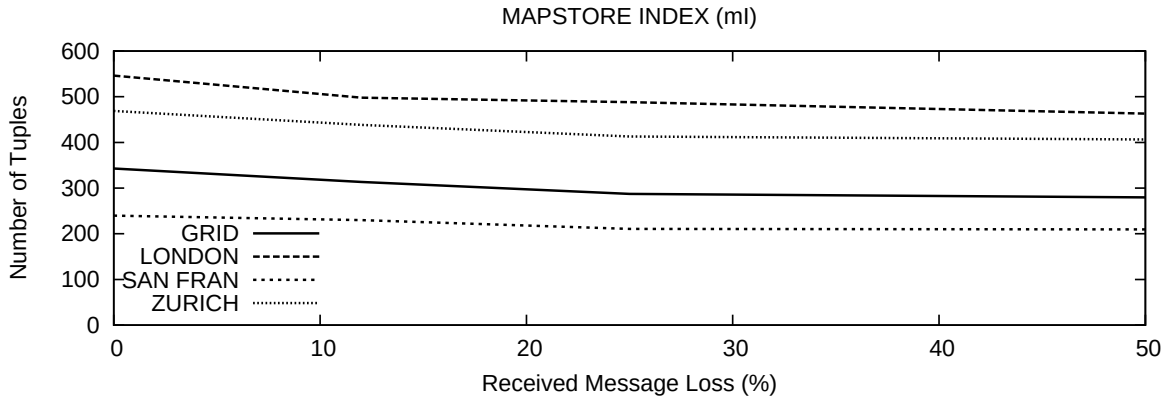


Figure 4.18.: Mapstore index (mI) tuples for increasing received message losses, after 1800 seconds of simulation time (POP=250, CR=200m, BP=12.5s).

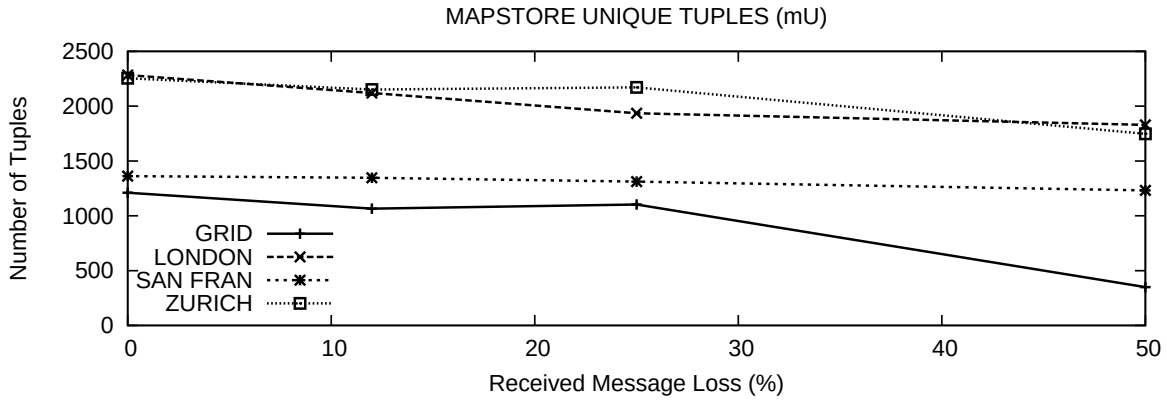


Figure 4.19.: Mapstore unique (mU) tuples for increasing received message losses, after 1800 seconds of simulation time (POP=250, CR=200m, BP=12.5s).

management and resources are necessary to share already shared data. However, the redundancy observed has a counter effect such that it makes the service more resilient to failure, and fragments have the possibility of existing for longer periods of time. Techniques to manage fragment selection may be applied to improve performance and make the service more efficient, for example, removing stale travel times from the service, only resending tuples a fixed number of times or limiting tuple hop counts.

4.8. Discussion

The Collect-Merge-Share protocol advocated in this chapter is lightweight. The estimation of travel times is an example of a "slow" decentralised vehicular service. To map the road network, vehicles share their own and others experienced mobility with neighbouring vehicles. Travel

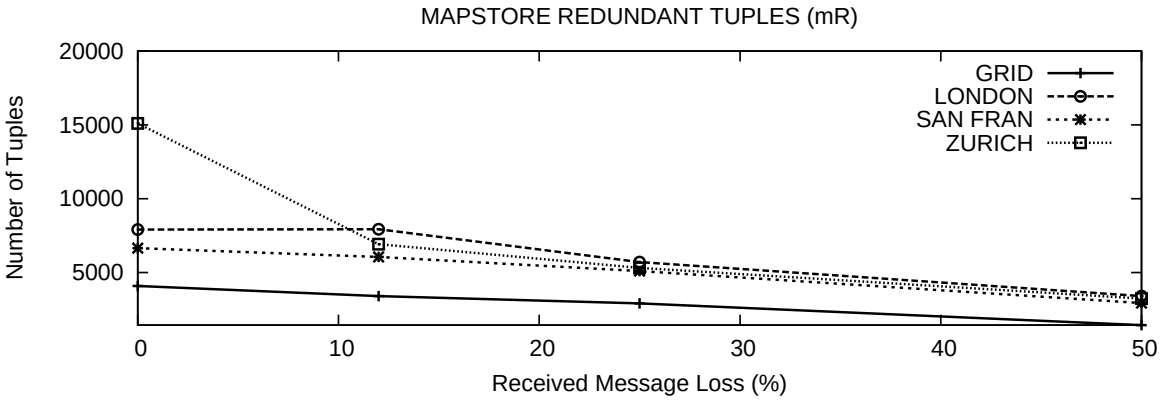


Figure 4.20.: Mapstore index (mR) tuple redundancy for increasing received message losses, after 1800 seconds of simulation time (POP=250, CR=200m, BP=12.5s).

time tuples disseminate slowly through the network. Travel time data sharing and processing is also not safety critical. Several trade-offs are associated with the usage of such a service. A core issue is the dynamic or changing nature of the state of the road network. Any estimation made by any vehicle will have some error associated with that estimation, as the road network is likely to change over time. The MapStore approach presented can be extended to include other data such as the number of neighbouring vehicles seen in a location. Further statistical data mining and network metrics can use raw travel time data collected to produce probabilistic results. Indeed, various centralised analysis methods can be applied to the MapStore of each vehicle.

Our experiments indicate that navigational maps already confine traffic to specific road sections more than others. There is no clear distribution, except in grid maps, of vehicle mobility over the entire map. This is visible in the heatmaps produced. It could be argued that present navigational routing techniques may be negatively affecting the road networks of various cities. This point raises the question: should all vehicles and drivers be provided with the same set routes when querying the same starting and ending positions? Such a question leads us to assessing the paradoxical effect of complete information about the road network, where all vehicles are assumed to know the state of every road section and travel time. With this information it could be argued that all vehicles will re-route themselves by what they deem to be the lowest cost time routes. Paradoxically, the system would move traffic to previously experienced road sections. The traffic and congestion would simply move to another road section and not be equally distributed over a collection of roads. The protocol

developed in this chapter has not assessed the feedback effect of vehicles re-routing using such maps; however, having incomplete information about the road network, is in this paradoxical case beneficial. One approach to combating this paradox with a centralised solution may be to randomise the routes provided to all querying vehicles. Centralised approaches to dynamic re-routing have proposed providing multiple routes to a driver or choosing a random route with similar properties from a collection of routes to a destination.

With such large road networks, it could be argued that computing routes in distribution is too computationally intensive and as such is better suited to processing away from the vehicle. A number of improvements have been made to handle large road routing [DSSW09]. Work by Sanders et al. has produced a method processing routes on mobile devices with minimal waiting time to the user [SSV08]. Even given these significant improvements - the majority of commuters in a city are unlikely to query extremely long routes (very often). Hence the actual storage capacity and analysis of the road network hierarchy may be sufficient, and storage and computational capacity already available on many vehicles may be more than required.

4.9. Conclusions

In this chapter we have investigated the feasibility and evaluated the performance of a new decentralised travel time estimation service. As vehicles travel they record their mobility in the form of trace histories. Trace histories are fragmented and shared with nearby vehicles to approximate a travel time estimation service which would normally be constructed using centralised approaches. Results show that such a decentralised service can approximate the operations of centralised service. A disadvantage of decentralisation is that travel time data sharing is not as immediate and mapped data is typically minutes old and fragmented, but approximates the operation of the road network. Data takes time to disseminate via the road network and data dissemination is highly dependent on the availability of vehicles. Increased vehicle populations and decreased broadcasting period improved service performance at the expense of messaging overheads. Decentralised vehicular services are capable of operating within the presence of data fault and failure due to the redundancy of data. Redundancy, however, requires data management else resource usage is inefficient. There is room for performance improvement using optimisation techniques. Travel time data is just one example of city data

which can be mapped. Various other stakeholders within the city may be interested in other sensory data, such as weather, pollution and road surface information.

The service considered within this chapter considered only messaging feedback. Also, the feedback cycle was typically long, operating of the period of 12.5 seconds or more is considered 'slow' feedback. In the next chapter we investigate a safety critical problem of decentralised intersection control, where feedback is required to be 'fast', such that vehicles may adapt their behaviour to correctly order themselves through an intersection.

5. Intersection Control

In the previous chapter we used the framework to develop a decentralised travel time estimation advisory service for drivers. The service used fragments of past vehicle mobility data and timestamps to associate recorded travel times with road sections and thereby map the road network. The service continuously remaps and modifies the current travel time view held by a vehicle such that a driver can be better advised on traffic conditions. As a service, travel time estimation operates over hours and is not considered time or safety critical. Within this chapter we explore the problem of intersection control. In contrast to the travel time estimation service, intersection control is a safety critical problem. The failure of the service could result in significant harm to the parties involved in any accident. The service operates at a shorter period than that for the travel time estimation service. The purpose of the development of a second service is to highlight the range of services that the framework can be applied to. Intersection control is an important and very difficult traffic management problem and has very different constraints and goals to the time estimation service.

The chapter begins by defining the intersection control problem and its patterns of mobility (Section 5.1). In Section 5.2 and Section 5.3 we outline related works, approaches and systems, as well as considering metrics used to measure the performance of road intersections. In Section 5.4 we detail the intersection scenario and Section 5.5 follows with our decentralised intersection control protocol. We evaluate the protocol in Section 5.6. We conclude the chapter with a discussion (Section 5.7), highlighting the trade-offs, outstanding issues and threats inherent in intersection control.

Two important contributions are made. Firstly we present, compare and demonstrate a protocol for decentralised intersection control. The protocol uses a set of collision avoidance techniques to direct vehicles through the intersection. Secondly, we compare the performance of the protocol to competing centralised traffic management approaches. Our protocol uses

ad-hoc messaging, collision avoidance and shared plans as a means by which to reduce delay, adapt a journey and maximise the efficient usage of a traffic intersection.

5.1. Problem

Traffic management literature [Lit09, GE68, Ino76, Hai63] defines an intersection as an 'at-grade' junction at which two or more transport *axes* cross at the same level. Within the context of a road network, these axes are roads. Defined alternatively, the road intersection represents the overlapping join of two roads. Vehicles intending to traverse the intersection are required to carefully navigate the intersection so as to avoid a collision with not only other vehicles, but often pedestrians and other road side objects (Figure 5.1). The traffic intersection problem is a useful micro-scenario from which to understand complex global traffic systems as the intersection often represents a bottleneck in the flow of vehicles (opening or closing the flow of a road section).

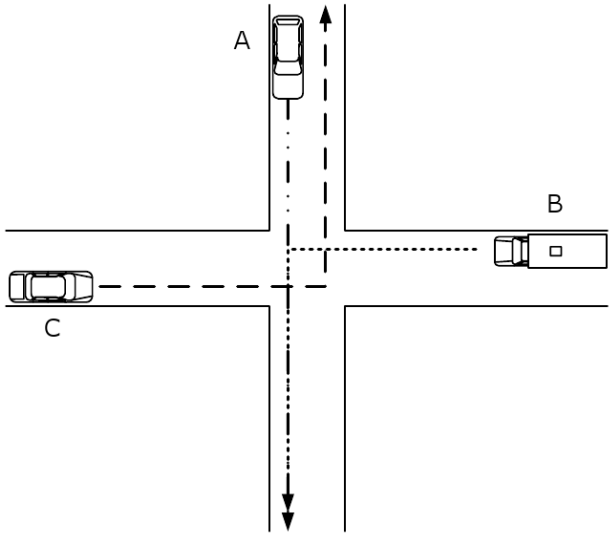
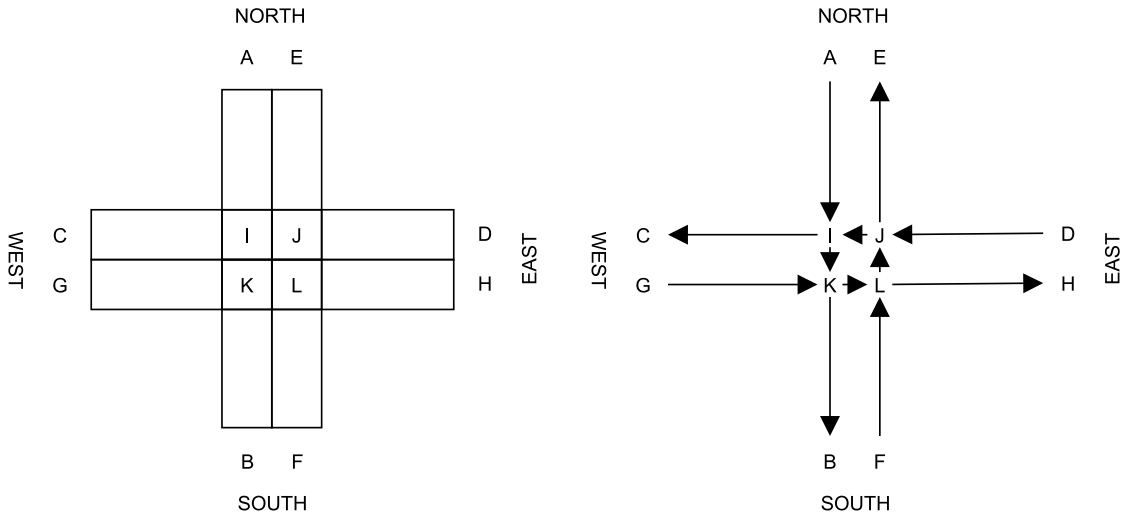


Figure 5.1.: Scenario: intersection with vehicles A, B and C competing to traverse the intersection in minimal time.

The definition of a road intersection used in this chapter is common to previous work by Baldessari et al. [LMP+07], Giridhar and Kumar [GK06], Hirankitti et al. [HKH07] and Dresner and Stone [DS04]. Two straight roads intersect one another at a perpendicular angle. There are two lanes with traffic flowing in opposite directions. The problem can be simplified

in processing terms. An intersection processes vehicles and the processing strategy used aims to direct vehicles flowing into the intersection, where an optimal outflow of vehicles would be a flow indistinguishable from the normal operation of an non-intersectioned road. The ideal, while perhaps unattainable, serves as an ever present goal towards which best efforts approaches can be used to achieve a sub-optimal flow strategy.

Figure 5.2 provides a representation of the intersection commonly used in related works. Letters denote significant positions and arrows depict the direction that vehicles should follow. The "intersection" is defined by the set of regions (labelled I, J, K, L). Vehicles can travel in lanes in opposite direction. Yet, where an intersection occurs, vehicles must be organised so that crossing vehicles do not collide. Traffic travelling from perpendicular angles is required to turn or cross existing flows to correctly navigate the intersection to travel onwards to one of three other goals in the subset of originating points. We should note that the natural organisation of intersection specifies that no more than two vehicles can collide at I, J, K or L. An example route can be defined as the set of points which a vehicle must reach, including those sub-positions, where a sub-journey may be specified by the route which uses $A \rightarrow I \rightarrow K \rightarrow B$. There are twelve possible routes to negotiate the intersection, formulated in three sets (Table 5.1). Using these routes we build traces which provide us with repeatable scenarios with which to evaluate the performance of varying protocols.



(a) Road intersection.

(b) Driving routes.

Figure 5.2.: An intersection: vehicles can enter and exit from one of four compass points. A vehicle can travel one of 12 mobility patterns.

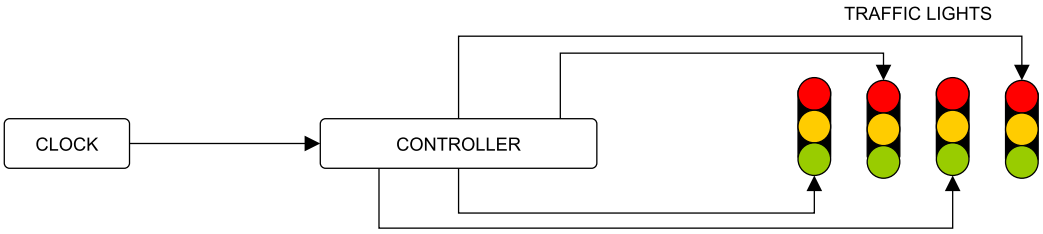
STRAIGHTS	TURNINGS	CROSSINGS
A → I → K → B	A → I → C	F → L → J → I → C
F → L → J → E	F → L → H	D → J → I → K → B
D → J → I → C	D → J → E	G → K → L → J → E
G → K → L → H	G → K → B	A → I → K → L → H

Table 5.1.: Intersection mobility patterns: the patterns represent all accepted vehicle routes and traversals of the 4-way two-lane intersection.

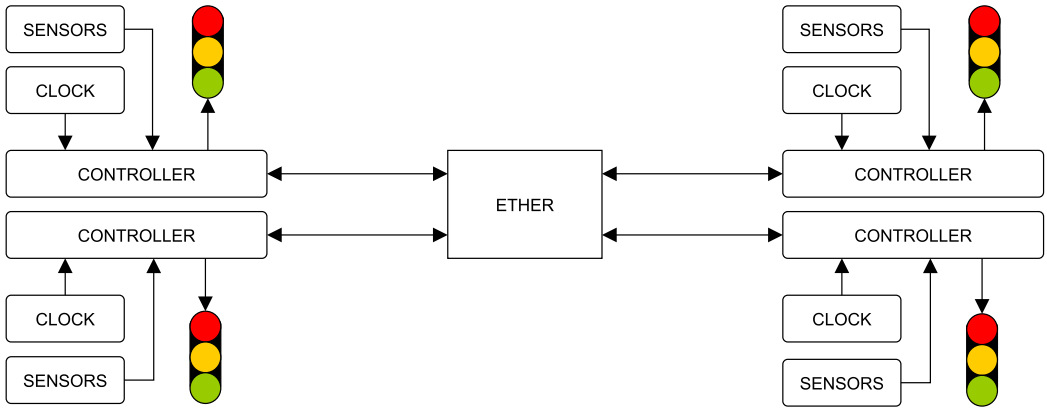
There are a number of motivating factors associated with achieving intersection control. Direct factors include safety (collision avoidance) and performance issues (maximising vehicle throughput). Vehicles traversing the intersection could collide if two or more vehicles attempt to use the same road network space concurrently. Collisions have the potential to kill. Secondly, performance aims to maximise throughput of an intersection to improve the rate at which vehicles enter and exit the system. However a number of secondary motivations include alleviating vehicle congestion in another region, reducing vehicle emissions, reducing experienced delay and improving the experience of driving. The intersection problem is predominantly one of flow control. An efficient intersection can be seen as an intersection which ideally does not hinder flow - however this is difficult. Roads leading to and leading out of an intersection are typically at right angles, requiring vehicles to slow to a manageable speed if turning or crossing the intersection. Vehicles which are turning or crossing the intersection require extra time to make turns safely. This slowing causes delays. To traverse an intersection, vehicles also require a following distance and gap by which they may cross. If no gaps are available then we see congestion occurring and thereby delay affecting vehicles.

5.2. Related Approaches

Road intersection control has a significant history of previous work. In the past, intersections were controlled without the aid of technology using generalised road rules. An example of this scenario is the four-way stop. All vehicles are required to stop before continuing their journey, decelerating and then accelerating, only continuing if having right of way. Other vehicles yield when the intersection is being used. A detraction of the four-way stop is that it turns the intersection into a single throughput queue, reducing the potential throughput of the intersection. The most widely adopted technological method to improve flow is the usage of



(a) Timed Traffic Control (TTC)



(b) Distributed Multi-agent System

Figure 5.3.: Related traffic light architectures.

traffic signals (for example via traffic lights). In certain instances, vehicles spend a significant amount of time waiting for a signal to change, even if no other vehicles are using the intersection at that time.

Most computer-based approaches to intersection control are centralised. While older traffic light systems used timing as a mechanism by which to control flows (Figure 5.3(a)), more recent methods have attempted to use agents to manage and operate traffic light control (Figure 5.3(b)). Proposals include Fuzzy Controllers [LLK99], Fuzzy Controllers that use Neural Networking [SKY99], Machine Learning and Classification approaches [BTAH04, MK94] and Multi-Agent approaches [HKH07, DMEP05, BCNR04]. Balmer et al. [BCNR04] consider agent-based approaches and the benefits of real-time observation on traffic flows and organisation, including the computational scale of the problem. They consider the procedure known as "4-step processing". Layers of computation and mobility are split into physical and mental layers. A feedback mechanism is used to "learn" behaviour. Strategies are generated and adapted to provide an increasingly improved strategy for each successive simulation executed.

In this way the model uses a period-to-period replanning approach developed in simulation to revise applied mobility strategies. Adaptation requires speed. If a system can adapt quickly to changes, then performance should not be detrimentally affected.

Hirankitti et al. [HKH07] presented a multi-agent approach utilising a single agent to control a set of traffic lights at an intersection. They demonstrated the usage of a rule-based approach for traffic management (simulated in NetLogo), defining the operation of an agent which continuously checks the status of the intersection and acting upon it as a *observe-think-act cycle*. Their approach considers traffic congestion is due to the improper traffic conditions that control traffic lights. Each traffic light is controlled by its own agent program and traffic lights. The approach seeks to reduce the delay time of vehicles at each junction. At each time-step, each agent controls all traffic lights at the road junction using an observe-think-act cycle. A supervisory agent is required. A large number of Condition-Action (CA) rules are generated, with 13 rules controlling the traffic-lights in different traffic conditions. In their results, the average-delay time is improved through the use of a collaborative approach.

In work by Doniec et al. [DMEP05], competitive multi-agent coordination is explored in the context of *deadlock* avoidance. The avoidance of deadlock is achieved using "anticipatory" methods, first defined by Rosen [Ros85]. The authors consider that the "observed traffic" is the "sum of all actors" and their actions. In our work, we consider the "sum of some" to be a better approach. Doniec et al. use various situation matrices to define the scenarios which an agent can find itself in. The matrices represent the crossing and interactions of two vehicles. SignalGuru [KPM11] uses mobile phones to identify and predict the traffic signal schedules of traffic lights, advising drivers on how they may maintain or improve traffic flow. Vehicles learn and schedule patterns with one another. SignalGuru predictions are reported to reduce vehicle fuel consumption by an average of 20.3%. Greenwave [GBT+09] proposes a distributed traffic management system which uses cameras and multi-agent algorithms to manage traffic intersection control.

Dresner and Stone [DS04] propose a reservation-based intersection control method, removing the necessity for physical traffic lights. However their requirement is that vehicles are automated and driver-less, such as those provided by the Google driver-less car initiative [Thr10]. The process of reservation begins when an approaching vehicle requests a slot from a central intersection manager (IM). The vehicle provides to the IM its estimated time of arrival at the

intersection and mobility data concerning its state and limitations (i.e. how quickly can the vehicle accelerate or decelerate?). The traversal of the intersection is simulated by the IM in relation to other vehicles also requesting use of the intersection. If a slot is not possible, the vehicle decelerates and is required to try again at a later time. The protocol developed does not consider physical issues relating to the limited movement of the vehicle. Moreover, multiple objects around the intersection may require slotting (e.g. pedestrians, parked vehicles). A key component is that vehicles should be autonomous. This is unlikely to happen in one step. This means that some vehicles would be driver-less and some would contain drivers. It is noteworthy that some of these problems are being considered by Google's driver-less cars initiative [Thr10].

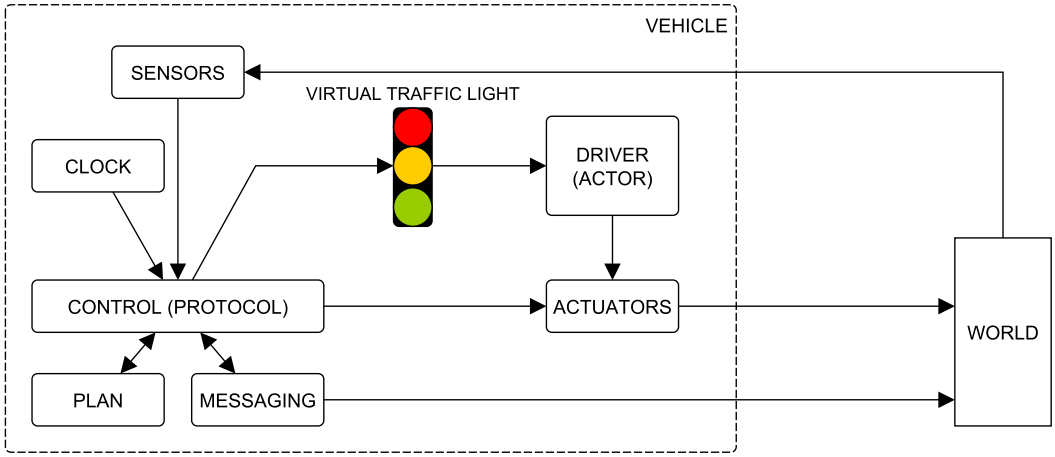


Figure 5.4.: Decentralised service approach: vehicles are advised uniquely (using virtual traffic lights).

In contrast to previous and current work, the protocol and algorithms described within this chapter consider a decentralised feedback-based and mobility sharing architecture (Figure 3.2). Where traffic lights do not exist, vehicles cooperate using messaging to safely organise and traverse a shared intersection. A virtual traffic light represents the advisory application to a driver (Figure 5.4).

5.3. Intersection Control Metrics

The usage of both *actuation* and *messaging* feedback warrants extended performance analysis. The intersection control service is measured using *message counts*, *vehicle throughput*, *travel times*, *delay counts*, *delay times*, *delay time-stamps* and *collision counts*.

Vehicle Throughput represents the rate of vehicles successfully traversing an intersection or road system, measured as the number of vehicles successfully exiting the intersection. Typically this is expressed in vehicles per hour. We sum the number of vehicles exiting the system at all four exit positions (using counts at each of the four exits North, East, South and West). Each individual vehicle is measured in terms of its route. On entry into the intersection vehicles hold an allocated specified *route* from the route set in Table 5.1. The specified static route is processed as a set of waypoints known as a journey *plan*. Each vehicle possesses a set pattern plan on how to use the intersection. Travel time (TT) is calculated for each vehicle and measures the elapsed time recorded from the time instant of a vehicle entry to vehicle exit (termed alternatively as a simplified end-to-end delay).

Estimated Travel Time is derived from the Estimated Time of Arrival (ETA). An estimated travel time is calculated using the displacement to the exit of the intersection, versus the maximum legal speed allowed. Vehicle delay time can be defined as the measured cost of adaptation (the extra elapsed time taken to travel a route). For each vehicle, delay is calculated as the difference between elapsed travel time and the estimated travel time taken to successfully travel through the intersection.

Delay is calculated as the difference between the ETA and the actual travel time. Delay counts, count the number of delay instances or times when a vehicle delays itself through action. The delay time refers to the time cost of these delay counts. Each vehicle route assumes that no delay shall exist along a route. An optimal journey is a journey without delay. Delay measurement is specific to the time and length of travel being measured - for instance a vehicle may be delayed and then make-up the delay (reduce the delay incurred) in subsequent time. Where delay times and counts are 0 we term a vehicle to experience "no delay". If delay is negative we term the travel time as 'gained' time, and thus a vehicle has operated beyond the optimal delay time expected. Mean delay is calculated, given a delay time sample space using the arithmetic mean.

Given a rate of vehicle input into the intersection from one of the four entry positions, vehicle throughput measures the mean rate of successful exit of vehicles using the intersection. Effectively, we measure the number of vehicles successfully exiting the road intersection at

each time instant. The delay count measures the number of occasions (delay instances) where a vehicle reduces its speed (brakes). Assuming a vehicle was not delayed, the vehicle would exit the intersection in optimal time. Delay time quantifies delays as a time cost. Where a vehicle comes within a minimum distance of another vehicle we deem two vehicles to have collided. While this is a metric, its existence identifies a protocol failure - collision is unacceptable. The aim is to minimise delay count and delay time, while maximising throughput and maintaining safety (without incurring no collisions between vehicles).

The Level of Service (LoS) rating is used by the traffic management industry to categorise traffic and congestion conditions over a particular road section, grading a road between best and worst grades - given favourable conditions (for example good lighting and weather conditions) [Lit09]. A best grade performance for a free road provides a throughput of 700 vehicles per hour per lane while a worst grade performance is measured beyond a throughput of 2200 vehicles per hour per lane. A problem with the LoS is that it considers speeds in excess of 48 kilometers per hour, where cornering vehicles are required to reduce their speed even lower to safely traverse the intersection.

5.4. Scenario, Aims and Assumptions

As previously mentioned, our scenario follows that presented in previous work [HKH07]. As a vehicle begins its journey, the driver of the vehicle inputs the intended destination of the vehicle into the navigational computer. A route is computed and a set of planned driving directions are provided to the driver. For a four-way intersection, four sets of vehicles may enter at each time instance from entry positions North (position A), South (position D), East (position F) and West (position G). We assume that no two vehicles enter at the same position at the same time instant (else a collision would occur between vehicles). Vehicles are separated by a following distance (often stipulated by driving handbooks as a two to three second delay [Mas06]). The aim of each vehicle is to drive a route from a entry position to a exit position without colliding with other vehicles. It is the aim of the intersection as a whole, to induce a minimal mean total delay to all vehicles using the intersection.

We assume that vehicles employ the vehicular framework architecture. As each vehicle travels within a minimum range of the intersection, the vehicle matches its plan to one of the

twelve available mobility patterns (Table 5.1). We also assume that vehicles are capable of communicating with one another via broadcasts using IEEE802.11p WAVE Short Messaging packets [IEE10]. There are no traffic lights visible to any of the drivers. Where a vehicle contains a driver, the driver is advised via a user interface (for example heads-up-display or audio prompts) on how to adjust her or his speed to avoid collision. The driver subsequently interprets these commands to adjust her or his mobility. We consider the probability of a driver following or disregarding a recommendation for adjustment. Where a vehicle is automated and without a driver, the vehicle automatically adapts its mobility to optimise the operation of the system. With multiple vehicles intent on using the intersection and the possibility of collisions occurring, vehicles are required to negotiate with one another to avoid collision and avoid an intersection reaching gridlock (a state of congestion where queues of vehicles prevent other vehicles from using the intersection).

5.5. Intersection Control Protocol

Our intersection control protocol uses shared state and mobility data to organise vehicles through an intersection. In the following section we begin by specifying the utility functions required by the protocol. These utility functions (Section 5.5.1) include methods for ranking, collision control and fault tolerance.

5.5.1. Utility Functions

Future Track

The *future track* (F), discussed in Chapter 3, is estimated from a mobility plan, is repeatedly calculated and refined every 100ms (as required by IEEE 802.11p WAVE). The constraints of the intersection scenario imply vehicles using an intersection have a limited number of means by which to traverse the intersection. This reduces the complexities of estimating a future track. F represents the set of estimated positions of a vehicle in future time-steps (T). Future tracks, a set of ordered waypoint-time pairs, are shared as a means of informing neighbouring vehicles of a vehicle's intended usage or goal of the intersection, a method of avoiding collision and ordering vehicles through the intersection.

$$\texttt{getFutureTrack(}R\texttt{.plan},R\texttt{.speed},T\texttt{)} \rightarrow F$$

We calculate the future track using the mobility plan provided by the mobility resource (R) as well as the current speed (the mobility pattern for the intersection). While the future track could conceivably be attached to a message payload, future tracks retrieved from neighbouring vehicles are calculated and expanded locally, a trade-off in favour of reduced message size versus increased computation. This is an implementation issue. Future tracks, once processed, provide an ordered sequence of waypoint-time pairs which predict the future mobility of a vehicle within a future time limit. Each future time instance is calculated. For instance, we may predict a set time ahead or a set distance ahead. For the purposes of the traffic intersection we calculate the future track for every position up until the vehicle has exited the traffic intersection area. A new future track is calculated for each cycle and the old version is discarded.

Collision

Our collision avoidance algorithm (Appendix C.1) determines the waypoint location (X) where two vehicles are estimated to collide. Compared future tracks identify three possible collision scenario, either (a) collision while *crossing*, (b) collision while *following* or (c) *no collision*. In (a) two vehicles contend for the same road section at the same time. In (b) a vehicle following collides with a vehicle ahead of it which it is following. The algorithm calculates X by comparing the future tracks (A and B) of two vehicles. Where A and B cross at a distance less than the specified minimum separation or sweep distance (D) we return the displacement distances of each vehicle (U and V) and the collision waypoint (X).

$$\texttt{collision(}A,B,D\texttt{)} \rightarrow [U,V,X]$$

This data is used by the protocol to advise the driver how to minimally adapt the speed of the vehicle to avoid a collision. These values are used by the ranking algorithm to determine the ordering of vehicles through the intersection. Choosing a separation distance which is significant enough to avoid collision with another vehicle is crucial. Notably the future tracks are provided by vehicles themselves. Hence, these future tracks represent the estimated positions of movement in future time-steps. We assume that future track estimations are accurate.

Cyclically rechecking for collision is required as the intersection is highly dynamic. As the intersection is a resource dependencies exist on its usage in scenarios of contention.

Ranking Condition

Where two or more vehicles approach the intersection the question that arises is - 'who has right of way? Who is allowed to use the intersection and in what order?' Road rules normally dictate this. For instance, vehicles generally yield to other vehicles approaching from the side of the driver. This presents some challenges to decentralised solutions. No traffic lights or centralised authority is directing traffic or stating when a particular vehicle may use the traffic intersection. To organise vehicle priority in the event of a predicted collision vehicles rely on a *ranking condition*. The ranking condition is a condition function which returns the deemed priority or rank of n-many vehicles - resolving which vehicle should use the intersection first. The ranking condition when provided the properties of n-many vehicles returns the identity of the vehicle which is given priority. For example, a ranking condition for two vehicles may be expressed as:

$$\texttt{ranking}(pA, pB) \rightarrow V$$

In this example, the ranking condition is provided a set of parameters (pA and pB) from two vehicles A and B. Ranking assumes that both vehicles know pA and pB. Parameters include speed, future track and perhaps other contextual data. We also assume the ranking condition used is known to both vehicles and that both vehicles apply the same ranking condition to the parameters provided. If this is true, then both vehicles A and B are expected to calculate the same result (V). Heuristically, if both vehicles have the same data, they can determine how other vehicles are expected to adapt and thereby adapt their mobility correctly. Within the context of the traffic intersection scenario, we consider a ranking using just two parameter sets.

Fault Tolerance

Intersections are expected to continue operating in the presence of messaging and/or sensory failure. The severity of a failure should elicit a response from a vehicle to take a safety critical decision. A number of failures could exist in the operation of the protocol. In a severe failure of a vehicle component a driver would normally remove herself or himself from the flow of traffic. For instance, in a tyre puncture a vehicle would park the vehicle out of the flow of traffic on the

side of a road. Similarly, we expect the same actions to be taken by a vehicle in the presence of a component failure or in the case of sensory failures.

Wireless communication can be unreliable. Intersection control and collision avoidance is safety critical. It is important that the protocol loop be capable of tolerating the loss of broadcast messages. The protocol executes every 100 milliseconds. While messages are being broadcast at a very high rate we improve the coverage of messaging by ensuring that messages are rebroadcast by neighbouring vehicles for a specified cycle count. Hence, if a vehicle fails to retrieve directly, it retrieves a message indirectly in the second cycle (from a neighbouring vehicle). Where only two vehicles are approaching an intersection this redundancy does not exist. Message rebroadcast and a low broadcast period is employed as a method to increase the chance of a message retrieval in the presence of failure.

5.5.2. Protocol

Our Intersection Control Protocol called the Vehicle Back-off Protocol (VBP) [BD10], attempts to maximise flow by providing vehicles opportunities to cross an intersection using gaps in traffic. The protocol strategy to maintain vehicle flow is motivated by work conducted by Helbing and Lammer [LH08] and centralised approaches like SCOOT [TFD98], as well as work by Reynolds [Rey87]. These multi-intersection adaptation methods, while considering a different scenario, highlight a maintenance of flow as a main goal for sequences of multiple intersections to perform effectively. The intersection in this context can be thought of as a switch - where the trade-off of improving flow in one direction is the starvation of flow in the crossing. Our VBP protocol is most similar to work by Ferreira et al. [FFCa$^+$10] where traffic lights are replicated virtually and presented in a customised form to the driver (vehicles use the intersection in customised traffic cycles). The work by Ferreira et al. does not consider the human driver within their system or consider what effect message faults may have on the operation of the system. Platoons of vehicles can still hinder flow in a particular direction if a crossing stream of vehicles is constant. The approach by Ferreira et al. is not purely decentralised and vehicles do not share mobility or negotiate.

Figure 5.5 shows two scenarios which the protocol is required to handle: (a) following (vehicles V1 and V2) and (b) collision scenarios (vehicles V2 and V3). A vehicle determines which adaptation to use based on the geographic location of a message and the contents of the future

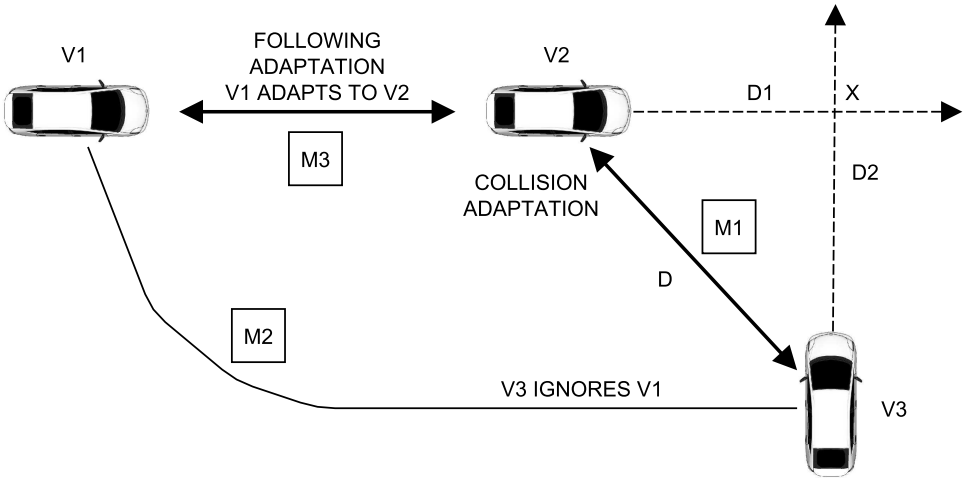


Figure 5.5.: Protocol adaptation scenarios. Vehicles (V1, V2 and V3) require adaptation to follow and avoid other vehicles while still ordering themselves to minimise overall intersection delay and maximise overall intersection throughput. V1 is following V2. V3 and V2 adapt with one another to avoid collision at X in a future time-step.

track. Messages are to prioritise adaptations.

The following scenario requires that a vehicle maintain a following distance which is significant enough to maintain both safety and flow. Adaptation should occur such that two vehicles avoid one another correctly without detrimentally affecting vehicle flow. The combination of scenarios can transmit contention. For example, if V3's adaptation requires that V2 reduce speed, then the distance between V2 and V1 shall close and V1 will be required to adapt (indirectly) to the actions of V3 (hence we complete a dependency between vehicles where V2 depends on V3, V1 depends on V2). The messaging link between V1 and V2 is labelled as M3. V1 and V2 broadcast state based data to one another due to broadcasts. The feedback element of the control is required to keep the state of all vehicles in a optimal state for best effort mobility. In the collision scenario, V3 and V2, if in contention, both calculate their collision point at X by using each other's future tracks. They calculate the distance between them and the collision point as well as the time to collision. Using a ranking condition the vehicles determine who has right of way and who must yield. If both V2 and V3 adapt in the same way, we would expect an accident to occur and therefore both vehicles in the VBP determine right of way by monitoring their speed, distance and time to X. If both vehicles are precisely the same distance away from X then vehicles act randomly to remove contention. The

more time a vehicle has to avoid collision the more minor incremental adaptation in mobility through feedback. A working alternative to random priority is the usage of vehicle identifiers to determine intersection traversal.

The protocol algorithm (Algorithm 2) leverages long range messaging to achieve this - vehicles adapt their mobility at extended distances to improve flow. In VBP, if two vehicles are contending for the same road space in the future, the vehicle furtherest from exiting the intersection reduces its speed. VBP considers the opportunity of applying strategies to improve cross flow by sharing vehicle future tracks as mobility fragments (a vehicle's intended traversal of the intersection). A trade-off of the approach is the possibility of gridlock if the intersection is overwhelmed by vehicles. Another disadvantage is that VBP requires 'gaps' or separation within traffic to allow for crossings. VBP uses the time before two vehicles meet to adapt mobility through the sharing of future tracks. The aim is that the overall operation of intersection be improved (to minimise delay and maximise throughput).

VBP employs both *short-range* and *long-range* strategies to avoid the probability of collision and thereby reduce the probability of having to substantially modify mobility (slow the vehicle to a complete stop). Messages are filtered by region and adapted to differently based on their existence within the separate regions (Figure C.1). A short-range strategy determines and adapts mobility if a vehicle is within the immediate vicinity. A long-range strategy adapts mobility for more distant orderings. Both strategies avoid collision (Algorithm 3).

For each cycle in the operation of the protocol, retrieved messages from the previous time-step are stored in the MI queue (Message Input queue). A *mobility resource* object (R) contains the journey plan of a vehicle and calculates a new future track (ft) at each cycle given the mobility plan of a vehicle (Figure 5.2 and Table 5.1). R provides an interface to position, direction and speed as well as other derivative framework functions.

Initially the protocol filters mobilities from neighbouring vehicles which affect the present journey plan using future tracks (filtering phase, lines 5 to 17). As described in Section 5.5.1, each vehicle estimates which other vehicles are set to collide with it at future time-steps. We consider two geographic retrieval areas to collect and filter messages. The first 180 degree orientation cone is used to gauge whether an object is ahead of another object (`isAhead`). A second cone of higher priority identifies the existence of vehicles directly ahead (`inFront`). We use the `isAhead` and `isCrossingTrack` to provide a first pass subset of future tracks.

Algorithm 2: Vehicle Back-Off

Input: A set of received messages MI, a mobility resource R, a pre-calculated partition distance pd, a clock t and a broadcast interval z

Output: A set of broadcast messages MO

```
1  begin
       // Filtering Phase
2      cI ← ∅
3      cE ← ∅
4      L ← ∞
5      foreach m ∈ MI do
           // Find closest short-range and closest long-range messages
6          if isCrossingTrack(m) then
7              Δb ← distance(R.position, m.position)
8              if (Δb < pd) ∧ (isAhead(m)) then
9                  cI ← m
10                 break
11             else
12                 if isCrossingIntersection(m) then
13                     [d1,d2,X] ← collision(R.ft,m.ft)
14                     if (d1 ≠ ∅) ∧ (Δb < L) then
15                         L ← Δb
16                         cE ← m
17     // Collision Avoidance Phase
18     if exists(cI) then
           // Short-range collision exists
19         reduceSpeed
20     else if exists(cE) ∧¬ exists(cI) then
           // Long-range collision exists
21         [d1,d2,X] ← collision(R.ft,cE.ft)
22         Δd = |d1 − d2|
23         Q = calcDisplacement(R.ft)
24         P = calcDisplacement(cE.ft)
           // Collision Avoidance Phase
25         if Q > P then
26             reduceSpeed
27         else if (Q = P) ∧ (priority) then
28             reduceSpeed
29         else
30             increaseSpeed
31
32     if (¬ exists(cI)) ∧ (¬ exists(cE)) then increaseSpeed
       // Sharing Phase
33     if t mod z = 0 then
34         nm ← newMessage(R)
35         MO.append(nm)
36     return MO
```

Messages are filtered into two closest message queues, closest internal (cI) and closest external (cE) using the partition distance (pd). A short-range strategy handles immediate collision threats and following distances. An `IsCrossingIntersection` function determines if the future track is the broadcasting neighbour leaving or entering the intersection.

The collision avoidance phase is priority driven (lines 18 to 32). The long-range strategy is typically applied first when no short range messages exist. pd is recalculated as the minimum separation distance with an additional two second variable separation (as is usually stipulated by law [Mas06]).

$$\texttt{pd} = \texttt{separation} + 2 \times \texttt{current speed}$$

The long-range strategy tests for a collision occurring in the future. This provides a [$d1$,$d2$,X] tuple if a collision is predicted. $d1$ represents the total step-wise distance calculated from the local future track (R.ft), while $d2$ represents the total step-wise distance calculated from the neighbouring vehicle and its computed future track stored within its payload. A position X represents the position where the two vehicles are expected to come within close proximity of one anther. The distances $d1$ and $d2$ are compared to provide Δd which if less than pd suggests to us that a gap in traffic is already taken and one vehicle must delay itself to follow behind the contended position.

The final phase of *sharing* (lines 33 to 36), periodically broadcasts a new message (nm) containing state data to all neighbouring vehicles. Neighbouring vehicles then are able to determine their own local adaptation in the next cycle. The message (nm) is placed into the MO queue (Message Output queue). Vehicles re-adapt every cycle towards an equilibrium goal of no-collision.

5.6. Evaluation

Our evaluation of the VBP considered more than 1440 experiments or 240 hours of protocol simulation time. Parameter settings were selected from transportation guidelines [Lit09, GE68, AS94, TvA01] and replicated for all vehicles entering the road intersection and each experiment was executed at least five times to ensure a sufficient sample of results. Using mobility patterns (Table 5.1), vehicles were injected (input) into an intersection at rates between 500 and 2000

vehicles per hour. The outcome of intersection control presents the throughput as an output rate (vehicles per hour) and the incurred delay as the experienced mean and maximum delays effecting vehicles. Hence we seek to maximise vehicle throughput and reduce delay without incurring collisions (increasing vehicle flows [LH08]). A collision would represent a safety critical failure on the part of the protocol. Our protocol performed, in most cases, better than comparable centralised approaches. We consider the trade-offs of the approach and highlight issues of gridlock in input rates above 1500 vehicles per hour.

Experiments sought to determine:

protocol messaging performance - the volume of messages sent, received and measure the number of messages acted on (Section 5.6.3);

delay and throughput performance in comparison to centralised approaches (Section 5.6.2);

identify scenarios in which protocol failure occurred (Section 5.6.4);

compare performance to existing intersection control approaches (Section 5.6.5);

identify trade-offs exhibited by varying parameters.

5.6.1. Setup and Parameters

The setup and parameters used to test the protocol were consistent with parameters used to measure *saturated traffic* [LH08]. We used results applied to individual intersection control strategies, including: *unsignalised*, *roundabout* and *timed traffic light controls* (60 second cycle). Each road vehicle was sized as 4.88 meters long and 1.8 meters wide. Each road was set at 6 meters in width. A 100 meter leadup distance exists to reach the intersection, with each intersection covering a 6 meter by 6 meter area. A maximum speed of 7 meters per second was used as this is deemed an appropriate approach speed for blind intersections [Mas06] (within 100 feet, approximately 30 meters from an intersection). For the first lead-up (100 meters prior to the intersection) a vehicle was required to reduce its speed to 2.5 meters per second. This slower speed allowed vehicles to make safe turns.

A limited 200 meter communication range was used. Intersection control payloads were short in comparison to message sizes considered in Chapter 4 comprising only 96 bytes. 0%,

Parameter	Value	Unit
Communication Range	200	meters
Broadcast Interval	0.1	seconds
Maximum Straight Speed	7	meters per second
Maximum Turn Speed	2.5	meters per second
Received Packet Loss (RPL)	0, 9, 12 %	packets
Vehicle Input	500, 1000, 1500, 2000	vehicles per hour (vph)

9% and 12% packet loss was applied to received messages [IEE10, Eic07] hence the protocol was simulated in the presence and without the presence of dropped packets. The vehicle input rate per hour was varied between 500 and 2000 vehicles per hour for all compared protocols and control methods. Vehicle rates equal to and above 1500 vehicles per hour exceeded the capacity of the road intersection. Experiments conducted beyond 2000 vehicles per hour were seen to overwhelm the intersection - vehicle throughput was too low and collisions between vehicles occurred. These vehicle input rates were chosen as upper rates were seen to saturate and overwhelm centralised methods using the same parameters.

Message payloads (Figure 5.6) contain a vehicle's geographic position (Waypoint), bearing, speed and future track data stored as a series of 64-bit floating point numbers (doubles). The protocol assumes that WSM packets are discarded immediately after use due to immediate message staleness. The future track, as presented in Chapter 3, represents the planned route a vehicle intends to drive and is stored as an ordered sequence of Waypoint positions. As a vehicle moves through the intersection, intermediary Waypoints in the future track are removed. When exiting an intersection a vehicles holds only the intersection end Waypoint.

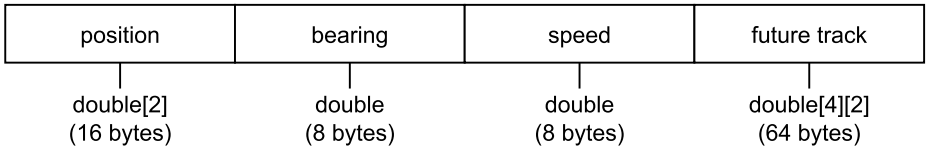


Figure 5.6.: VBP payload.

5.6.2. Delay and Throughput

Ideally, intersection control attempts to minimise the mean delay experienced by each vehicle using an intersection - resulting in a maximisation of throughput. Centralised traffic manage-

ment approaches typically produce cyclical delay. In contrast, VBP is typically seen to produce irregular patterns of delay and throughput. This is visible by comparing the total instances of delay and total throughput occurring between varying approaches. Within the patterns is we should be aware of the period of patterns and the number of patterns occurring within an elapsed time interval.

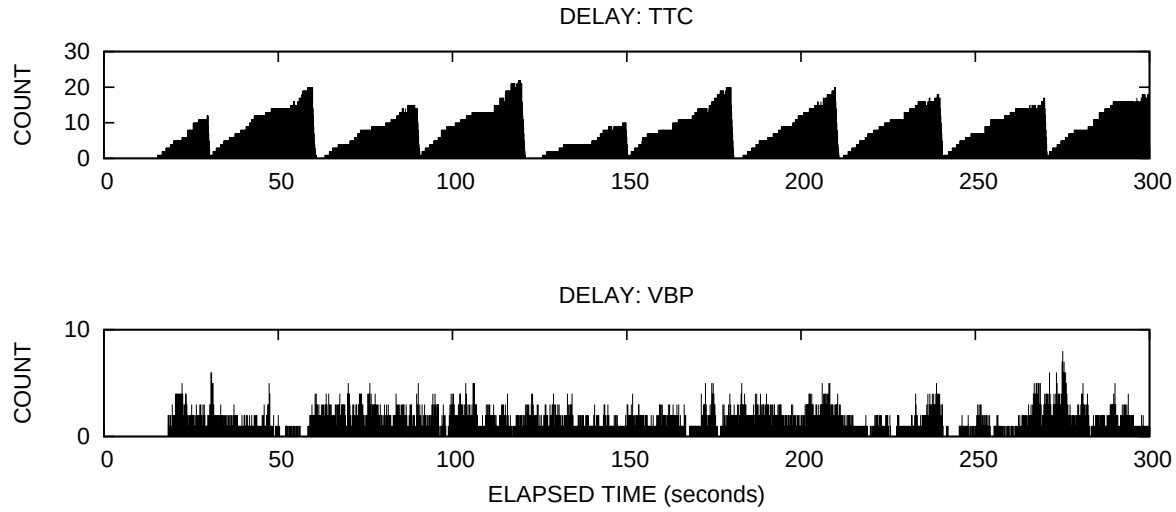


Figure 5.7.: Delay: Comparison of TTC and VBP delay instances for a single experiment, given a 300 second sample.

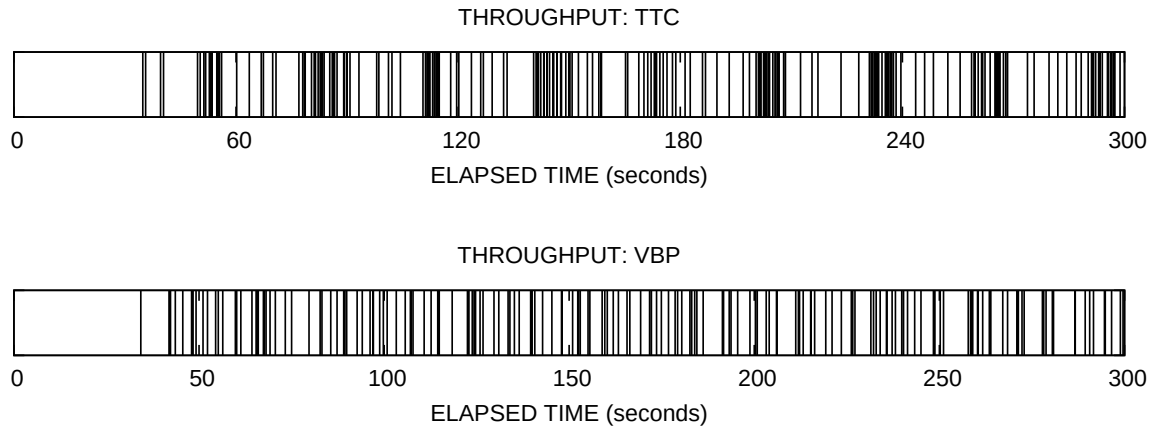


Figure 5.8.: Throughput: Comparison of TTC and VBP throughput instances for a single experiment, given a 300 second sample.

Figure 5.7 shows the typical total instances of delay occurring at an instant of elapsed time (the metric approach is used by traffic authorities [GE68, Hel01]). Timed Traffic Control (TTC) exhibits both cyclical and larger units of total delay instances. Interpreting the patterns

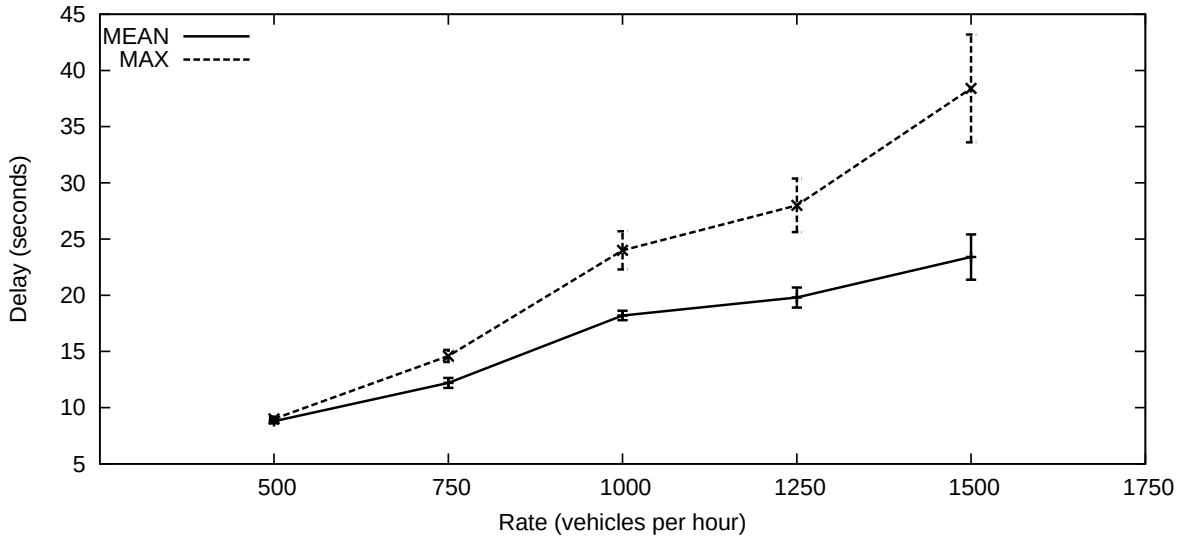


Figure 5.9.: VBP delay performance for varying intersection input rates (vehicles per hour). Delay is seen to grow linearly for increasing vehicle input rates. However, as the vehicle rate is increased, the deviation of delay is also seen to increase. This is coupled with an increasing divergence between maximum and mean delays.

of delay and delay tells us about the behaviour of vehicles using the intersection. On the intersection itself, 'saw' shaped delays are seen as vehicles backed up behind one another. TTC throughput (Figure 5.8) shows a similar effect, with groups of vehicles exiting the road network at just prior to 60, 120 and 240 seconds. These grouped patterns of throughput are a sign of platooning vehicles. In contrast, the total delay instances experienced with VBP are smaller and more randomly dispersed due to slow adaptation on part of vehicles as they approach the intersection (Figure 5.7). VBP delays are an order of magnitude smaller than those of TTC. VBP throughput is more irregular, with gaps existing between consecutive throughput instances (Figure 5.8). The grouping of vehicles is less severe and platooning does not typically occur on the scale seen in TTC.

VBP delay (Figure 5.9) and throughput (Figure 5.10) performance for successive vehicle input rates was seen to be linear. In Figure 5.9 we compare the mean and maximum delay experienced by vehicles for flows between 500 and 1500 vehicles per hour, with error bars describing the standard deviation of results for successive experiments. Figure 5.10 shows the actual vehicle throughput versus the set vehicle input rate. As expected, the throughput (output) for an intersection is seen to perform below the input rate. The diagonal input rate represents ideal intersection throughput. The performance of a service is hence measured by

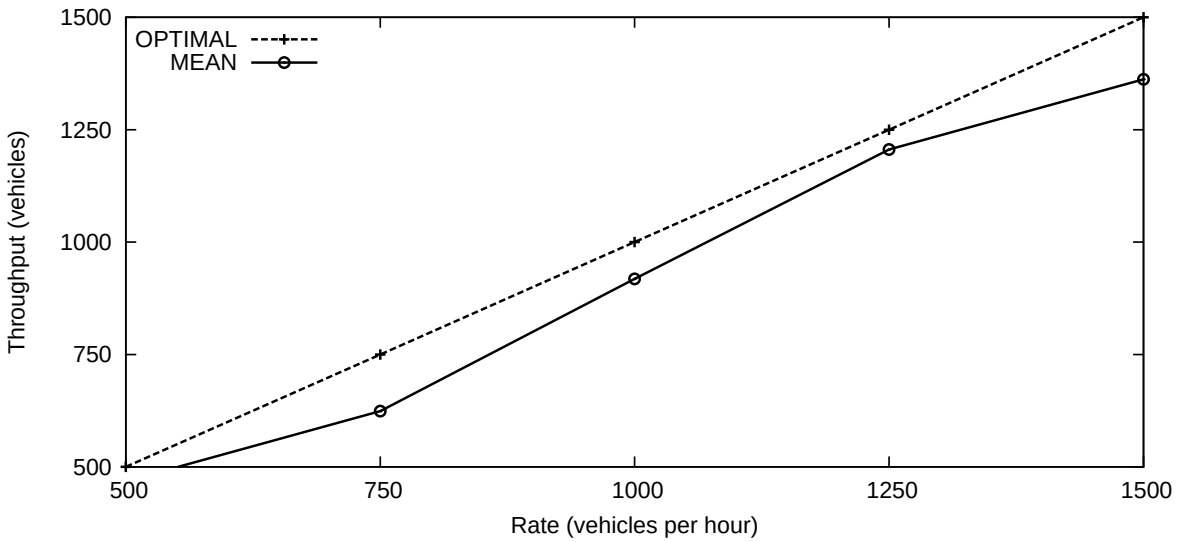


Figure 5.10.: VBP throughput performance for varying intersection input rates (vehicles per hour).

how close the throughput performance comes to this ideal.

5.6.3. Messaging and Adaptation

Message broadcast is dependent on the broadcast period specified (Section 5.5.2). For individual scenarios message growth was at first exponential and then linear, where exponential message growth was observed in the initial start-up of the intersection (new vehicles entering the empty intersection). The number of vehicles existing within the intersection was shown to affect the marginal increase in broadcast messages for a given elapsed time. Similarly, message retrieval is directly dependent on broadcasts, hence message retrieval counts grow as a multiple of the number of vehicles within proximity of the broadcaster.

Table 5.2 shows protocol performance for varying vehicle inflows (vehicles per hour) and packet losses. The table presents the mean number of message broadcasts (B) and receives (R) counted per minute, as well as the mean number of adaptations (A). The R:B ratio highlights the redundancy of messages and the multiplying effect of broadcast given the population of vehicles attempting to traverse the intersection. The A:B ratio focuses on the number of adaptations given the number of broadcasts made.

Broadcasts multiply the number of messages received by neighbours, a consequence of the broadcasting action. We assume that drivers would react to advice by adapting their speed as advised by the VBP protocol. The linked adaptations of vehicles highlighted the dependencies

between vehicles (Appendix C.3). Message retrieval was seen as exponential for increasing vehicle inflows. We see a direct increase in the number of messages received and the number of adaptations (speed up or slow down) made by vehicles to reorder themselves. This is expected as the vehicle population within the intersection increases, thereby increasing the number of broadcasts. Also, an increase in vehicle density increases the probability of contention or collision and thereby increases the likelihood of required avoidance by adaptation. We see the exponential growth the ratio of adaptations versus broadcasts (A:B) common to all results for varying packet loss.

(a) No packet loss.

Rate	Broadcasts (per minute)	σ	Receives (per minute)	σ	Adaptations (per minute)	σ	R:B (ratio)	A:B (ratio)
500	3556	13	18024	154	99	11	5.07	0.03
750	4914	32	37857	369	274	24	7.70	0.06
1000	7764	197	95702	5193	1432	247	12.33	0.18
1250	8934	1001	141105	16628	2046	262	15.79	0.23
1500	12133	240	254769	12072	3496	235	21.00	0.29

(b) 9% received packet loss.

Rate	Broadcasts (per minute)	σ	Receives (per minute)	σ	Adaptations (per minute)	σ	R:B (ratio)	A:B (ratio)
500	3562	22	18071	246	87	14	5.07	0.02
750	4949	8	38280	140	292	15	7.73	0.06
1000	7006	33	74922	580	863	15	10.69	0.12
1250	9219	127	133063	4247	1951	153	14.43	0.21
1500	11803	370	224829	14367	3212	273	19.05	0.27

(c) 12% received packet loss.

Rate	Broadcasts (per minute)	σ	Receives (per minute)	σ	Adaptations (per minute)	σ	R:B (ratio)	A:B (ratio)
500	3373	12	16118	125	82	8	4.78	0.02
750	4926	14	37714	175	289	35	7.66	0.06
1000	6958	36	68874	806	794	100	9.90	0.11
1250	9175	224	130356	5803	1949	270	14.21	0.21
1500	12940	1879	275111	100168	3529	1085	21.26	0.27

Table 5.2.: Message counts and adaptations made, for varying input rates (vehicles per hour) and varying packet loss, where σ represents measured standard deviation.

A major consideration for vehicles using the intersection is the increase in messages. At minimum inflows, each vehicle is expected to handle on average approximately 6 messages per execution cycle. In contrast, at maximum vehicle inflows, approximately 20 messages exist per

execution cycle. Hence it is required that WAVE Short Message (WSM) hardware be capable of processing and queuing these messages readily to reduce the likelihood of protocol error. In this case, retrieving a message too late could have a detrimental effect on the performance of the intersection and the safety of vehicles.

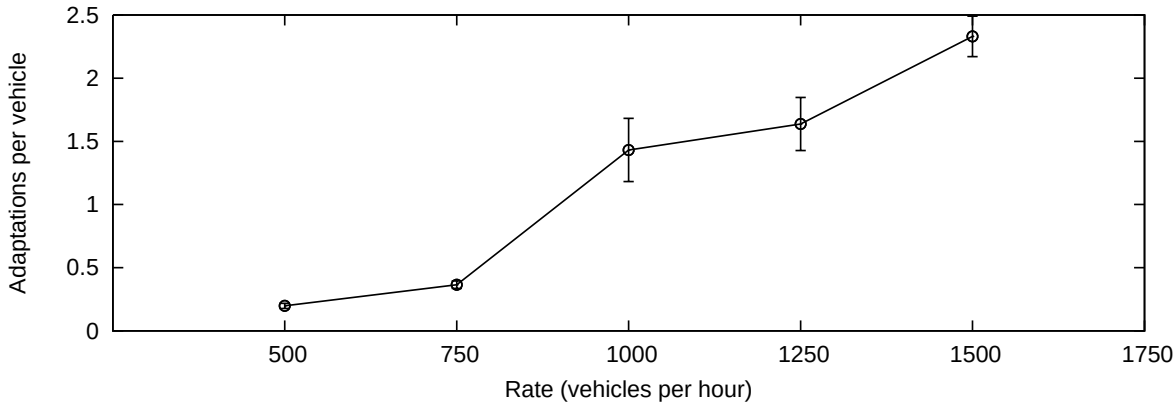


Figure 5.11.: Number of adaptations per vehicle for increasing vehicular input flows (vehicles per hour).

Figure 5.11 shows the mean number of adaptations made by vehicles for changing flows. For less dense intersections, 500 and 750 vehicles per hour, the low requirement of adaptations means that some vehicles do not need to adapt to the changing vehicle flows. Between 750 and 1000 vehicles per hour there is a sharp rise in the requirement for adaptation, from 0.36 to 1.43 adaptations per vehicle. Again from 1250 to 1500 vehicles per hour we see a marked increase from 1.63 to 2.33 adaptations per vehicle.

Increased packet loss was seen to both reduce received messages and directly reduce the number of adaptations occurring. When comparing 0%, 9% and 12% packet loss, the A:B values were reduced for associated vehicle flows. For example, A:B was 3% of messages for a vehicle rate of 500 vehicles per hour with no packet loss, while A:B was 2% of messages for a vehicle rate of 500 vehicles per hour with 9% packet loss.

5.6.4. Dropped Packets and Deadlock

To simulate failure, 9% to 100% of retrieved messages were dropped for flows of between 500 and 1500 vehicles per hour. Figure 5.12 presents a collision matrix showing the mean number of collisions experienced by vehicles where varying the dropped packet counts between 0% and 100% and increasing vehicular flow between 500 and 1500 vehicles per hour. The matrix

highlights the probability of collision given varying conditions. For example, we should expect collisions to occur at vehicle rates of 500 vehicles per hour where the packet loss is in excess of 70%. Similarly, we find that the protocol is likely to fail for 9% message loss at vehicle flows of 1500 vehicles per hour.

As was consistent with message dependency, failure in messaging caused vehicles to collide and hence the protocol was seen to be less than robust in certain scenarios, where vehicles could not communicate at the current time-step. A possible solution to this would be to have vehicles retain a model of the intersection in store. Where a vehicle had no data about the position, speed or bearing of another vehicle, errors were made in calculating the appropriate action required to avoid collision - hence vehicles collided. Such collisions typically took place due to failure within the protocol short-range scenario handler (where the time and distance between vehicles was already limited by proximity).

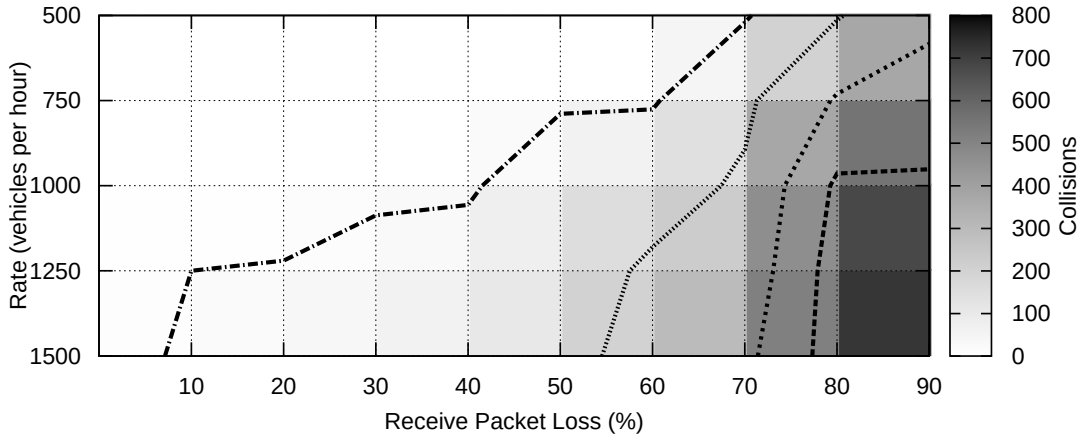


Figure 5.12.: Collision matrix maps the number of collisions occurring for increased numbers of dropped receive packets and increasing input flows (vehicles per hour).

Where adaptations were not correctly managed and vehicle densities are significant, VBP performed less well leading to gridlock occurring at the intersection. Within a deadlocked position, vehicles from all four input roads cannot pass through the intersection due to inter-locking needs. A vehicle may be attempting to cross, but is blocked by a second vehicles. In turn the blocked vehicle blocks another vehicle and so on. This locking makes the intersection unusable. Hence the protocol fails to operate in certain instances where vehicle input rates are above 1500 vehicles per hour. VBP as a protocol is unable to identify or resolve deadlock

scenarios automatically as it is concerned with ordering rather than deadlock resolution. In certain intersection scenarios, deadlock is unavoidable give particular vehicular behaviour.

5.6.5. Performance Comparisons

VBP performance is directly comparable to single intersection scenarios, including *unsignaled*, *roundabout*, *timed traffic management* (TTC) and *actuated timed traffic management* (ATTC) strategies. For comparison, our TTC and ATTC results used a 60 second split cycle (30 seconds red and 30 seconds green), where ATTC assumes the presence of inductive loops at each of the four approaches to the intersection. The decentralised protocol is not directly comparable to the SCOOT [HRBW81, TFD98], Sydney Coordinated Adaptive Traffic System (SCATS)[Low82], Greenwave [GBT$^+$09] and Lammer et al. [LH08] algorithms as all consider linked road sections and uneven traffic inputs from specific input roads. These approaches prioritise traffic flow along a single axis. SCOOT purports a maximum 30% overall improvement to traffic flow for connected intersections.

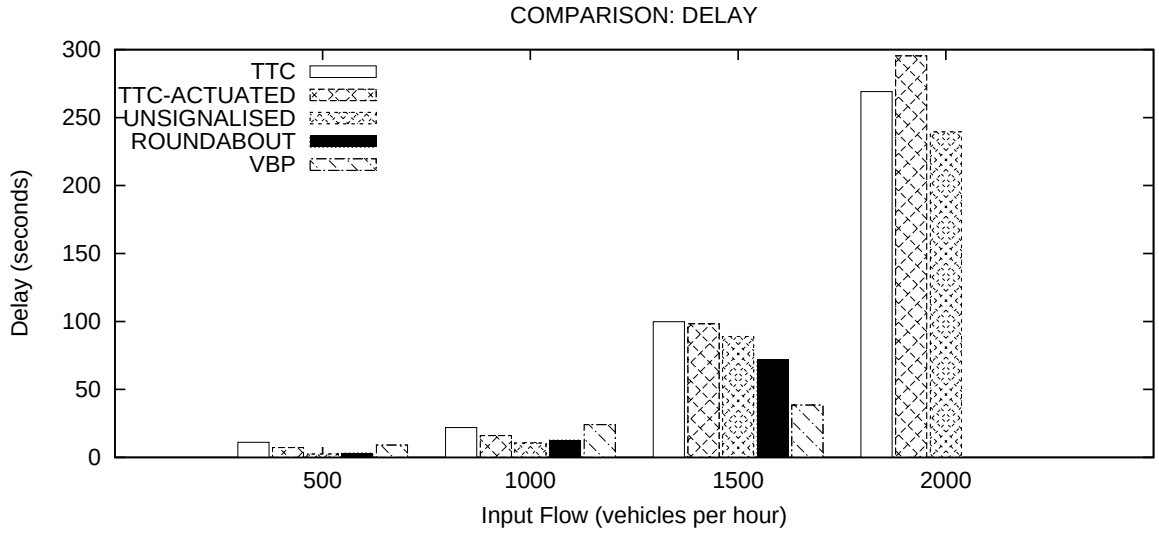


Figure 5.13.: Comparison of delay per vehicle between Timed Traffic Control (TTC), Actuated TTC, unsignalised, roundabout and VBP for varying vehicle input rates between 500 and 2000 vehicles per hour. A lower delay value is preferable.

In simulation, mean VBP performance was similar to alternative approaches for intersections handling up to 1500 vehicles per hour. Within scenarios with flows greater than or equal to 1500 vehicles per hour most VBP simulations failed in deadlock after less than 588 seconds of operation. In comparison to centralised approaches, VBP had varying performance benefit,

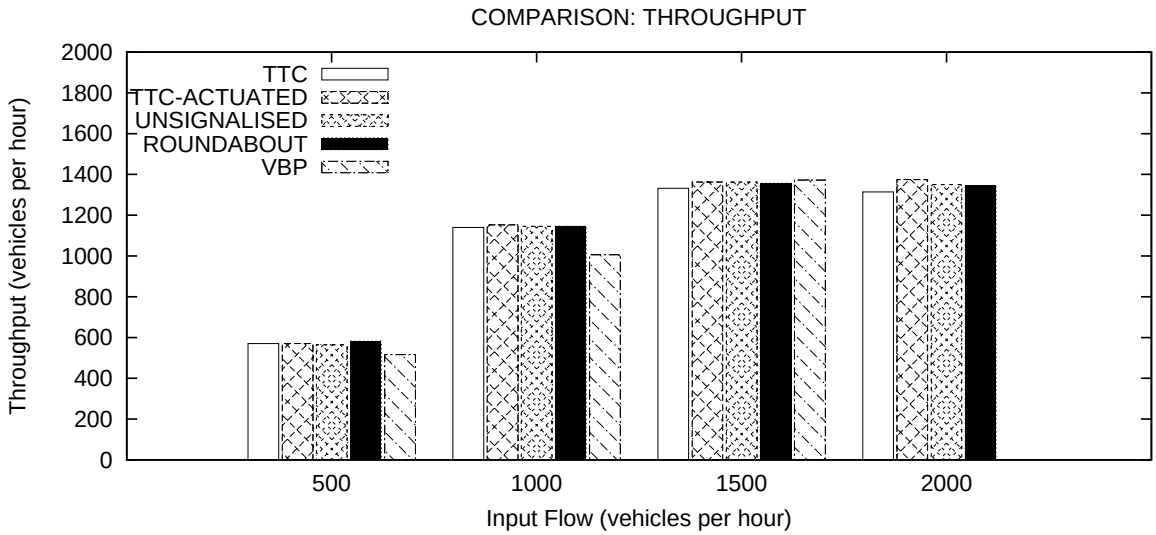


Figure 5.14.: Comparison of throughput between Timed Traffic Control (TTC), Actuated TTC, unsignalised, roundabout and VBP for varying vehicle input rates between 500 and 2000 vehicles per hour. A higher throughput is preferable.

showing suitability for particular vehicle rates, however the VBP strategy failed for higher vehicle inputs. Figure 5.13 shows that VBP delay was minimised in comparison with all other approaches for vehicle populations of between 500 and 1500 vehicles per hour. Vehicular throughput (Figure 5.14) exceeded all other approaches with VBP throughput for 1500 vehicles per hour was 1372 vehicle per hour or a 91% input to output success.

Performance benefit is dependent on delay or throughput priorities. Notably, roundabout performance failed in scenarios where vehicles flow from a no yielding position. Vehicles stopped and yielded flow for long enough to starve flow and cause congestion. Hence, roundabouts are useful in scenarios where the largest traffic flows are originating from a non-yielding position. Similarly VBP was only successful in scenarios where vehicles made enough space for other vehicles to cross and where enough spacing was maintained. An altruistic incentive is required to exist for VBP to operate. This is not the case in roundabouts - yielding is specified by road rules. Hence both VBP and roundabouts have their trade-offs.

The performance of the intersection is predominantly affected by the inter-relationships between flow, distance, messaging and safety. Intersection performance is finite provided a following distance and physical limitations of a vehicle and its occupants. An intersection can only physically accommodate and process a limited number vehicles. Similarly, the strategy used to maintain flow for an intersection is affected by the rate at which vehicles enter the intersection.

Figure 5.14 shows the performance drop loss where input rate exceeds the limitations of the VBP strategy. Experiments with rates beyond 1500 vehicles per hour showed the protocol to fail to maintain sufficient throughput and collisions were more likely to occur. At certain vehicle input rates, vehicles soon saturated the intersection beyond capacity. A subsequent effect of this was that the number of messages communicated within the intersection rose significantly. This presents the problem that if vehicles become densely populated within a region, they have the capacity to reduce the effectiveness of decentralised services operating within the vicinity.

VBP is seen to reduce the mean delay affecting all vehicles. However, the intersection experiences a throughput 'short-fall', seen in Figure 5.14. This throughput perfromance is a result of speed controls employed by a VBP vehicle as it enters an intersection. For example, where a vehicle approaches the intersection from A to I or G to K (Figure 5.2). Prior to intersection traversal, vehicles maintain a following distance. However, instead of some vehicles lowering their speed at the intersection, all vehicles lower their speed. Traditional approaches do not require a safety speed as a vehicle crosses I, J, K or L in the intersection. Vehicles entering the intersection are thereby moving more slowly than vehicles in alternative approaches. Lower throughput is a consequence of this initial slow approach to the intersection and the interleaving of vehicles to reduce mean delay. Exiting the intersection, vehicles accelerate to the legal maximum speed specified. Results show that decentralisation is feasible, VBP is however lacking as it does not outperform traditional approaches in terms of throughput.

5.7. Discussion

Ideal traffic intersection control seeks to reduce the delays experienced by vehicles, while also maximising vehicle throughput [HRBW81, TFD98, Low82, GBT+09, LH08]. VBP attempts to improve vehicular flow such that such flows may reduce mean delay and improve throughput for varying inflows (vehicle rates entering the intersection). Maintaining correct flow is highly beneficial. Yet, traffic intersection control is a bounded problem. Where contention occurs between two or more vehicles, one or more vehicles must typically adapt or yield their mobility for the benefit of the system as a whole. A vehicle must trade-off between delay and safety for the maintenance of throughput. The intersection itself is limited physically by its capacity to hold vehicles. If a vehicle does not yield or order itself, we see reduced flow on specific input

roads and in a worst case scenario we see deadlock. The required adaptation problem is well suited to decentralised feedback driven solutions.

However, collision avoidance is a safety-critical concern and a primary challenge. The type of collision avoidance used in VBP is more complex as it combines spatial and temporal mobility data to communicate more complex mobility patterns between vehicles. VBP represent a strategy to maximise the usage of an intersection. Within the problem of intersection control, delay may be classified as a secondary aim and as such timed traffic management systems attempt to partition and allocate time either according to a regime of "fairness" or specifically to demand (for example inductive loops and the employment of sensors). Importantly these control systems have safety critical elements (orange lights) or periods in which the intersection can be reset to deal with a new collection of crossing vehicles. VBP attempts to maximise the usage of the intersection by focusing on collision avoidance and adapting speed to "nudge" the operations into a state of minimal delay. VBP then attempts to maintain this state of operation within the intersection. The example given in this chapter may be considered a simplified model of traffic intersection control, yet it provides a base comparison on which further work can be applied. A more realistic approach may take into account the added complexities of such an intersection, including pedestrian mobility, weather and road conditions. However we should still be clear as to measure system performance in terms of collision, delay and throughput. As public traces for intersection usage were not available, we used synthesised mobility traces. Real intersection traces would more accurately describe and measure an intersection for presently used protocols; however, as vehicles adapt their mobility pure traces would only provide more realistic start and end points for given intersections. There is room for improving the VBP to more intelligently handle changing technical and environmental conditions occurring within the intersection domain. For example, we have not considered emergency scenarios, where vehicles might obstruct traffic flow, but still be part of the intersection control system.

The results presented illustrate the difficulties traffic intersections pose to vehicular flows. Trade-offs exist in terms of safety, delay, throughput and speed. In terms of decentralised control, one of the significant problems with collision avoidance lies with positioning and future track estimation. Vehicles require an extremely accurate measure of where they are, how fast they are moving and where they are estimated to be in the future. Improvements to the future track estimation method are possible, for instance providing the future track estimation

algorithm with more contextual information about the environment and nearby vehicles. The data collected at intersections by the service not only facilitates intersection control, but also presents a means by which city administrators and mapping systems can observe the flows occurring at particular intersections.

5.8. Conclusions

In this chapter we have described and evaluated a new decentralised intersection control protocol which we have named the Vehicle Back-off Protocol (VBP). Within limits and for specific parameter settings, VBP is seen to reduce mean delay in comparison with some presently available centralised intersection control approaches. A significant issue with the protocol is that it produces large numbers of messages which, in large densities, could overwhelm present technologies and their capability of retrieving and computing within the short time intervals necessary to ensure safe intersection control. As an advisory service, VBP requires that vehicles have highly reliable inter-vehicle communication but that they also act rapidly to adapt to changes made by neighbouring vehicles using the intersection. Human drivers are unlikely to make these adaptations quickly enough.

The VBP protocol and service illustrates both usage of the framework to construct decentralised traffic management services and the potential for decentralisation within the context of intersection control to improve flows. Notably, VBP does not require the deployment or maintenance of traffic lights. Hence, there are significant opportunities to improve the flow of vehicles using a single two-lane intersections where vehicles may be aware of one another over larger distances (perhaps kilometers rather than meters).

6. Conclusions and Future Work

Previous and current computing approaches to traffic management have almost exclusively focused on centralised control systems. While such approaches improve service availability, and data access times, and allow for authoritative control, their usage comes at great cost in terms of setup, deployment and maintenance requirements. The inclusion of V2X technologies within vehicles, as part of Intelligent Transportation Systems (ITS), enables us to develop alternative decentralised approaches which leverage vehicular interactions, mobility patterns, feedback and scale as a means by which to construct and provide services which mimic centralised services. This mimicry, while seen to be feasible, requires the consideration of a number of trade-offs of which some are scenario specific. Yet, such decentralised services are complex to develop as vehicles are both mobile, dynamic and messaging. The framework provides us with a tool by which we may develop and evaluate decentralised traffic management services.

6.1. Summary

Within the thesis we began by outlining and motivating decentralisation. Thus far, traffic management solutions have come at significant cost with most solutions being centralised and monolithic in their design. The advent of V2X technologies in vehicles allows us to consider decentralised alternatives to reduce the costs of such traffic management systems.

As V2X frameworks and tools are lacking, we addressed the issue of providing a geodetic scaffolding framework for the development and evaluation of decentralised vehicular services. Decentralised vehicular services represent and alternative means by which to construct traffic management systems. They are characterised by their: (a) decentralised ad-hoc operation, (b) use of wireless broadcasts, (c) use of feedback control loops, (d) sharing of mobility data, (e) inter-vehicle cooperation and (f) adaptation. Our case studies used positioning (spatial) and

time (temporal) data to derive more complex functions. Dynamic mobility fragments were used to specify mobility. Fragments were shared to enable vehicles to adapt to one another for service provision.

In contrast to other approaches (which typically use packet switching standards) and packet switching overlays (for example IPv6), our framework uses short broadcast messages modelled on WAVE Short Messages [IEE10], part of the Vehicle-to-X (V2X) communication standards.

In Chapter 2 we considered the state of the art proposals motivating vehicular services, which consist predominantly of Intelligent Transportation Systems (ITS) and Smart City initiatives. While many application proposals have been made, few concrete systems have been successfully implemented. We described the related work affecting the architecture of the framework as well as simulation frameworks which have been used to simulate vehicular ad-hoc networks (VANETs).

In Chapter 3 we presented a new geodetic vehicular framework for decentralised services. The framework sought to address the requirements of mobility specification, decentralised control, cooperation and scalability, fault tolerance and extensibility. We did not consider issues of trust, privacy or incentives within such services.

The framework consists of a layered world model, comprised of mobility and messaging overlays which are modelled in the Geographic Urban Simulator (GUS). The GUS is an integral part of the framework, allowing us to develop, simulate and evaluate prototyped decentralised services. The framework provides an architecture and abstraction for service construction. Services are built from protocols that are provided messaging, time and sensor inputs over successive execution cycles. Inputs are processed by each service protocol and at the end of each cycle both actuation events and/or message broadcasts occur. The framework and individual protocols are feedback driven and broadcast-based, using collected and shared mobility fragments.

Vehicles communicate with one another using ad-hoc gossip style messaging. Protocols are short and imperative. Most significantly services share and interpret modifiable geodetic mobility fragments (past, present and future) to enable many services. Feedback control loops are used to repeatedly adapt the operation of services within a set of equilibrium goals - improving their operation through successive interactions (both actuation and messaging). A consequence of leveraging the interactions between vehicles is that the code for services is

written more simply. A trade-off is that systems are limited in their ability to adapt due to communication ranges. Service code, once written, is simulated in detail using the GUS, prior to deployment, to demonstrate and validate behaviour. Prototyped services are easily portable to mobile devices such as Android platform devices.

To demonstrate the use of the framework, we explored the behaviours of decentralised vehicular services by developing and simulating two traffic management services: (a) a decentralised *travel time estimation service* and (b) a decentralised *intersection control service*. Each service, while using similar components, is a unique sub-scenario of traffic management.

In Chapter 4 we described a distributed travel time estimation service which seeks to map changing travel times to road maps. The service provides a means by which to assess traffic and congestion on the road network over time. Vehicles collect and share their experienced mobility as travel time tuples (mobility fragments). The service uses a slow feedback cycle and periodic broadcasts to monitor, disseminate and adapt travel time data held about the state of the road network. We successfully demonstrated how such a decentralised service would operate in the context of failure, and considered the benefits of such maps and data to both drivers and city administrators. Data was visualised to build heatmaps. Maps constructed were often unique to a particular vehicle as a vehicle's data was a function of the contacts which had occurred in previous time-steps.

The service requires no central authority for its operation. As a trade-off data availability was slower. Vehicles were dependent on contacts and community sharing for mapping. By increasing the vehicle population we improved service performance and availability. The availability and redundancy of data proved to support service operation even in the context of limited failure; however, a trade-off of this success was that both messaging and processing resources were somewhat inefficient and message management was required to optimise the advantages and disadvantages of data redundancy. Travel time data is just one example of data which can be mapped. Various other stakeholders within the city may be interested in mapping other sensory data, such as weather, pollution and road surface information. Future work could use the protocol and service developed to dynamically re-route vehicles through the city road networks - in an attempt to avoid traffic or heavily used sections of the road network. Such a dynamic re-routing service would attempt to solve the complete information problem discussed in Section 4.8.

In Chapter 5 we described a new decentralised intersection control service. In comparison to the travel time estimation service, the approach use a faster feedback cycle that broadcast messages every 100 milliseconds (the limit of WSM broadcasting). Using messaging and modifications in mobility, vehicles approaching an intersection cooperated using a control protocol called the Vehicle Back-off Protocol (VBP) to avoid collision and order themselves through the intersection. The protocol was seen to operate well for minimal packet loss scenarios. The approach failed where packet loss exceeded 9% and vehicle inflow rates (vehicles entering the intersection) exceeded 1500 vehicles per hour. Successful VBP performance approached the performance of a roundabout without the negative characteristics of halting individual feeds of traffic, as is the problem in roundabouts. While the decentralised approach is feasible, messaging requirements exceeded the limitations of present IEEE 802.11p WAVE technologies. Furthermore, the approach requires that human drivers react and adapt beyond their ability to adjust vehicle speed instantaneously to avoid collisions and maximise gaps in traffic. While the intersection service would fail in reality, the protocol points to gains which could be achieved if vehicles ordered themselves over extended distances.

6.2. Similarities, Limitations and Trade-offs

Comparing the costs of centralised and decentralised approaches is difficult. There are hardware and software costs, communication costs, costs of installation and maintenance, costs of compliance to environmental and safety standards, costs of insurance and legal liability. Furthermore costs can be incurred by many parties, infrastructure providers, by those in charge of road planning and maintenance, by vehicle manufacturers, by software/service providers, by drivers. A proper comparison would require a rigorous approach to modelling such costs. This is complicated by the fact that V2X and other technologies are evolving very quickly and any particularly economic model or technical solution might be superseded before it has a chance to establish itself.

Several similarities, limitations and trade-offs exist between centralised and decentralised vehicular services. Both centralised and decentralised services depend on vehicles to collect and share data. The more vehicles existing on the road network, the more sensory samples available and the more probable the likelihood is of vehicle interaction - given an implied

density of vehicles and the constraints which road systems make on vehicles. Centralised service data is stored in a logically central location (central authority). Vehicles consult a single and known authority to access services. Data is potentially highly available with time bounded access. In contrast, decentralised approaches sample data in a particular geographic region of a vehicular network, yet this data takes time to disseminate. Hence, data is typically older in comparison to centralised data stores. However, decentralised services do not suffer from issues of failure as centralised approaches. As decentralised services are scaled up they require increased management for the organisation and control of data.

Centralised services depend on a mobile data network and authority to transport data. Interactions with the central authority can be initiated via point-to-point data streams (for example IPv6), as long as mobile data connectivity is available (within cities this connectivity is almost always available except in regions of bad coverage). While mobile data networks are available, the addition of millions of cars may place considerable pressure on their performance. Indeed, it would cost mobile data networks more if centralised services were used by all. As vehicular services are position-based, both approaches assume highly accurate measurement, approximation and positioning methods. Without accurate data, services could suffer from failure with the outcomes effecting vehicle safety and performance. While a decentralised service may not deliver instantaneous service, we have shown that data dissemination can communicate data within minutes over short distances.

Both centralised and decentralised services require a critical mass of vehicles to cooperate with one another for services to become effective. Without vehicles or infrastructure to sense the road network, service provision is not possible. Furthermore, the data collected and shared must be relevant to the locality in which vehicles intend using the service. It is left to the protocol developer to determine what data is more relevant given message positions, payloads, recency, hop counts and contacts. For example, a vehicle collecting data about New York is not particularly useful to data being used by a vehicle in London. The value of data can be considered from the point of view of sensory input and relevance. A vehicle might be measured broadly by the accuracy and number of samples provided. For example bus networks due to their regularity are likely to provide regular data about specific parts of a road network. Yet, similarly, the data provided is more relevant to some vehicles than others.

Both centralised and decentralised vehicular services need to be scalable. Decentralised ser-

vices do not require the improvement of computational capacity or the provision of a networking infrastructure as found in a centralised client-server infrastructure. For instance, central authorities may act as bottlenecks and as single points of failure. Decentralised vehicular services are more resilient to network failure in terms of their independence from mobile phone networks. Given a densely populated road network, when a vehicle is no longer part of the vehicle community we lose valuable sensory data, but we do not lose the overall service (once again this is dependent on the density of vehicles). In dense vehicular networks the redundancy of data, due to broadcasting, improves service resilience. As we reduce the vehicle population, resilience decreases. However, in sparse vehicle communities, data is not capable of persisting unless vehicles re-enter the road network. An example of this scenario is vehicles exiting the road network at the end of a day of travel. Many cities see their road networks unused overnight while populations sleep. Hybridising vehicular services to use both WAVE standards and mobile data networks (as well as static road-side objects) may have significant benefits.

6.3. Discussion

Previous chapters have demonstrated the feasibility of building decentralised services in software as well as the methods of measuring management methods. WAVE messages and their integration into vehicles and road-side infrastructure will allow them to message one another to build decentralised services. Strong motivation for V2X use comes from industry [Sie11, LMP+07, Thr10, Bar09]. The future openness of V2X standards is questionable. For instance, it is unlikely that safety critical applications, such as collision avoidance and intersection control systems, should be accessible to all developers. Legal requirements will likely dictate what will be acceptable in terms of automation and service access.

A limitation of vehicular services demonstrated within the thesis, is that we assume that vehicles are altruistic. In reality, vehicles and drivers may not be altruistic, however services such as Waze [Waz11] are examples of communities which act altruistically to improve service performance. However, a question arises as to whether vehicles would share beneficial data with other vehicles in a setting where sharing that information would affect their quality of service. For example if vehicles are competing for road resources would they share information. This problem refers back to the problems of complete information - where it should be noted

that the physical limitations of vehicles existing in the real world make acting on all complete information more challenging. For example, a vehicle may have complete information about the city, but be unable to act to benefit itself because of, for instance, being too far away from an unused road.

6.4. Future Work

The work presented in this thesis offers a great deal of scope and opportunity for future work. A large number of challenging problems still exist within the vehicular domain where solutions could have significant benefit in reducing the costs associated with traffic management systems. We divide future work into short-term and long-term challenges. Short-term challenges reflect those challenges which might be immediately considered by using the framework. While long-term challenges encompass challenges which require the adoption of real ITS technologies and standards.

6.4.1. Short-term Challenges

While shown to be feasible in simulation, the protocols and services developed using the framework, make a number of simplifying assumptions and would benefit from a number of optimisations. Travel time estimation could benefit from both data aggregation and geographic data selection techniques. Data mining techniques might be applied to travel time data collected over extended periods of time to explore traffic patterns over extended time intervals. Similarly, the VBP could benefit from storing and estimating extra positioning data about the past and expected future locations of neighbours. VBP does not consider the effect of realistic pedestrian usage on control, which is expected to make control more complicated. This remains a challenge to the protocol.

One of the objectives of our framework is service portability. While services have been written in Java as imperative feedback loops and simulated, the thesis has not investigated the actual deployment of a protocol onto a set of real vehicles. Such work would serve to calibrate framework results and metrics in relation to a real world deployment, measuring the quality of the Geographic Urban Simulator (GUS). Indeed, one of the initial intentions of the GUS was that it provide a means by which to rapidly port protocols to Android devices. A significant

issue related to testing is the capability of scaling such real systems to the scales necessary to correlate simulation performance with real world scaled systems. The need for simulation is motivated by the fact that even in the context of real systems, testing and evaluation will continue to be required prior to deployment. It would be infeasible to re-deploy test services on vehicles repeatedly due to time constraints. Rather, simulation of a service would help to build confidence in an approach prior to deployment testing.

An outstanding challenge is the consideration of the complete information problem. Given a mapping application, it may be possible for all vehicles to know at all times the traffic and density affecting all roads within the city. This complete knowledge of the road network presents a problem if all drivers act to use the 'best' route to reach a destination. The problem can be distilled as a centralised scenario. Given two roads, where one is free of traffic, this may motivate all vehicles using a congested road to use the empty road, thereby moving congestion to the free road and rendering the previously congested road, free of traffic. The effect of dynamic routing using a decentralised vehicular service is not known. Simulating such systems may be a way to build up a better understanding of how such systems would effect driver behaviour in reality. The GUS is well suited to exploring this problem.

The focus of the thesis has been on the provision of vehicular services that only use vehicle-to-vehicle communications. Preliminary results for hybrid architectures that use vehicles, road-side nodes and/or remote cloud-based servers point to improvements in the dissemination of data as well as the control of vehicles in specific scenarios [WBT+10][Waz11].

6.4.2. Long-term Challenges

A number of new applications such as pollution monitoring, population density monitoring, dynamic re-routing, safety systems (for example accident detection) and city maintenance systems are potentially well suited to using decentralised approaches. The travel time estimation application presented could be modified to operate with train services. Largely disconnected train systems like the London Underground could benefit from such an approach, where trains and passengers would share mobility data. Operators and travellers would benefit from shared estimations concerning the state of the underground rail network. The roll out of such technology might be easier in comparison with the cost of deploying new sensors and technologies deep underground and across such an expansive train network.

A selection of works have considered VANET and automotive security problems and methods [MNBJ11, KCR+10, CMK+11, AB11], as well as privacy issues [QWDFZ11]. Security and privacy challenges remain. In the context of intersection control or other safety critical applications, the opportunity to break the system presents a significant threat (even in non-safety critical scenarios). Privacy issues exist in the sharing of location and trace data. Using vehicular traces an attacker could infer information about where a person may live or work. Such data also presents data holders with a method of profiling drivers - for example, insurance companies may be interested in the number of kilometers a driver is driving and where they drive. The usage of such data could increase premiums for the driver. The distribution of a vehicular service in a decentralised form does allow individual vehicles to use pseudonyms to increase levels of anonymity. For instance, pseudonyms could be used for as long as a service is being used. A new pseudonym could then be generated when using the service again. However, a trade-off of anonymity is that it reduces accountability - vehicles which attempt to break the system are difficult to remove from the service as we cannot identify these misbehaving vehicles using identifiers.

6.5. Closing Remarks

The traffic monitoring and control domain is a non-trivial problem domain and developing systems has long been a challenging issue. Obvious solutions do not exist to a multitude of traffic management problems. For many years traffic management has been the focus of research for many mathematicians, physicists and engineers. In the long run, solutions may only be capable of reducing traffic inefficiencies within limits. This is partly because road networks have a limited capability and capacity to enable the flow of vehicles from location to location. While security, trust, privacy and incentives are challenges to the adoption of vehicular services, a number of other orthogonal issues remain unsolved. They present varying barriers to the adoption of decentralised vehicular services in the context of both mapping and intersection control, as well as other applications. For both applications, a critical mass of adoption is required to enable decentralised mapping to occur. If the set of vehicles using travel time estimation services are too small, then services cannot be provided. In ther context of intersection control, lack of adoption may result in lives lost. Outside the technical challenges

exist legal challenges to the operation of autonomous or semi-automous vehicles. The legal libility of whom is to blame in the event of a failure of a service is questionable. Alternatively, the legal ramifications of automation may require that all vehicles are automated.

For the immediate future, human beings are likely to remain the core supervisory component in vehicles; as such, vehicles don't always behave in a predictable way. While there have been many successes in automating driving [Thr10], it may be some time before computers take complete control over driving, if ever. In the interim, we are likely to see automation and adaptive computing used in particular scenarios, for example vehicle platooning, safety stopping and automated parking. It is perhaps not unlikely that the trend of automation in vehicles may be similar to that seen during the adoption of aeroplane autopilot technologies. Autopilots within aviation became increasingly complex and today the pilot is often irrelevent to the flight and landing of an aircraft.

A number of companies have started to consider decentralised ITS services [Sie11, LMP+07, Waz11, Thr10, For11]. The adoption of a collection of proprietary centralised services suggests that vehicular services are now practicable, affordable and beneficial to drivers. Decentralised vehicular services offer alternative approaches to centralised approaches. While there are trade-offs to decentralised services, their usage has the potential to reduce costs, improve road efficiency and provide city users and city managers a means by which to harness and leverage vehicles to sense the city, process this data and improve city operation. A hybridisation and merging of centralised and decentralised service characteristics and architectures may be beneficial to a number of traffic management problems.

Bibliography

[AB11] Tansu Alpcan and Sonja Buchegger. Security Games for Vehicular Networks. *IEEE Transactions on Mobile Computing*, 10(2):280–290, February 2011.

[ABFeH07] S. Ahmed, M. Bilal, U. Farooq, and Fazl e Hadi. Performance analysis of various routing strategies in mobile ad hoc network using QualNet simulator. In *Emerging Technologies, 2007. ICET 2007. International Conference on*, pages 62 –67, nov. 2007.

[AM08] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA, 2008.

[AS94] Richard Arnott and Kenneth Small. The economics of traffic congestion. *American Scientist*, 82(September-October):446–455, 1994.

[Bak09] S.C.G.U.A. Bakshi. *Feedback Control Systems*. Technical Publications, 2009.

[Bar09] Dave Barth. The bright side of sitting in traffic: Crowdsourcing road congestion data. http://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html, 2009. [Online; accessed October-2011].

[Bat07] M. Batty. *Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals*. MIT Press, 2007.

[Baz07] Ana L. C. Bazzan. Traffic as a complex system: Four challenges for computer science and engineering, 2007.

[BB06] A.R. Beresford and J. Bacon. Intelligent transportation systems. *Pervasive Computing, IEEE*, 5(4):63 –67, oct.-dec. 2006.

[BCNR04] Michael Balmer, Nurhan Cetin, Kai Nagel, and Bryan Raney. Towards truly agent-based traffic and mobility simulations. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 60–67, Washington, DC, USA, 2004. IEEE Computer Society.

[BCSW98] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, New York, NY, USA, 1998. ACM.

[BD09] Rudi Ball and Naranker Dulay. Approximating Travel Times using Opportunistic Networking. In *2nd IEEE Intl Workshop on Opportunistic Networking*, May 2009.

[BD10] Rudi Ball and Naranker Dulay. Enhancing Traffic Intersection Control with Intelligent Objects. In *First International Workshop the Urban Internet of Things 2010 - Programming the real-time city.*, December 2010.

[Bet01a] Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:55–66, July 2001.

[Bet01b] Christian Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '01, pages 19–27, New York, NY, USA, 2001. ACM.

[BGJL06] John Burgess, Brian Gallagher, David Jensen, and Brian N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking. In *Proceedings of IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[BHT+03] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott,

and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128–136, June 2003.

[BHvR05a] Rimon Barr, Zygmunt J. Haas, and Robbert van Renesse. Jist: an efficient approach to simulation using virtual machines. *Softw., Pract. Exper.*, 35(6):539–576, 2005.

[BHvR05b] Rimon Barr, Zygmunt J. Haas, and Robbert van Renesse. *Scalable Wireless Ad hoc Network Simulation*, chapter 19, pages 297–311. CRC Press, August 2005.

[BMJ+98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM.

[BPC+07] Paolo Baronti, Prashant Pillai, Vince W.C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7):1655 – 1695, 2007. Wired/Wireless Internet Communications.

[BRCMGCRH08] J.C. Burguillo-Rial, E. Costa-Montenegro, F. Gil-Castineira, and P. Rodriguez-Hernandez. Performance analysis of ieee 802.11p in urban environments using a multi-agent model. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1 –6, sept. 2008.

[BTAH04] L Bull, A Tomlinson, Jd Addison, and Bg Heydecker. Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In *In L. Bull*, pages 276–299. Springer, 2004.

[CB05] David R. Choffnes and Fabián E. Bustamante. An integrated mobility and traffic model for vehicular wireless networks. In *Proceedings of the 2nd ACM*

international workshop on Vehicular ad hoc networks, VANET '05, pages 69–78, New York, NY, USA, 2005. ACM.

[CBD02] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, pages 483–502, 2002.

[CBH+01] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss. Interplanetary internet (ipn): architectural definition, 2001.

[CCC+06] Liwei Chan, Jirung Chiang, Yichao Chen, Chianan Ke, Jane Hsu, and Haohua Chu. Collaborative localization – enhancing wifi-based position estimation with neighborhood. In *Links in Clusters, Proc. International Conf. Pervasive Computing (Pervasive 06*, pages 50–66, 2006.

[CCL+11] Francesco Calabrese, Massimo Colonna, Piero Lovisolo, Dario Parata, and Carlo Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):141–151, 2011.

[CD96] Gordon D. B. Cameron and Gordon I. D. Duncan. Paramics: Parallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10:25–53, 1996. 10.1007/BF00128098.

[Cha99] Xinjie Chang. Network simulations with OPNET. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 307 –314 vol.1, 1999.

[CMK+11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[CRK08] Francesco Calabrese, Carlo Ratti, and Kristian Kloeckl. Wikicity: Real-time location-sensitive tools for the city. *Handbook of Research on Urban Informatics: The practice and Promise of the Real-Time City*, 2008.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[DMEP05] Arnaud Doniec, Ren M, Stphane Espi, and Sylvain Piechowiak. S.: Dealing with multi-agent coordination by anticipation: Application to the traffic simulation at junctions. *In: EUMAS*, 2005:478–479, 2005.

[DMP+10] Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 63–76, New York, NY, USA, 2010. ACM.

[DS04] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pages 530–537, Washington, DC, USA, 2004. IEEE Computer Society.

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*. Springer, 2009.

[DSW06] A. G. Dimakis, A. D. Sarwate, and M. Wainwright. Geographic gossip : Efficient aggregation for sensor networks. In *5th International Symposium on Information Processing in Sensor Networks (IPSN 2006)*, Nashville, TN, April 2006.

[EDH+96] Thomas Ewing, Ezzat Doss, Ulf Hanebutte, Thomas Canfield, Alenka Brown

van Hoozer, and Adrian Tentner. Argonne simulation framework for intelligent transportation systems. http://www.osti.gov/bridge/servlets/purl/219342-RDvURk/webviewable/219342.pdf, April 1996.

[EGH+08] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 29–39, New York, NY, USA, 2008. ACM.

[Eic07] S. Eichler. Performance evaluation of the ieee 802.11p wave communication standard. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 2199 –2203, oct 2007.

[Enk03] W. Enkelmann. Fleetnet - applications for inter-vehicle communication. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 162 – 167, june 2003.

[EP05] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[Fal03] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[FFCa+10] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K. Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, VANET '10, pages 85–90, New York, NY, USA, 2010. ACM.

[FGH+06] Richard M. Fujimoto, Randall Guensler, Michael P. Hunter, Hao Wu, Mahesh Palekar, Jaesup Lee, and Joonho Ko. CRAW-

DAD data set gatech/vehicular (v. 2006-03-15). Downloaded from http://crawdad.cs.dartmouth.edu/gatech/vehicular, March 2006.

[FHFB07] Marco Fiore, Jerome Harri, Fethi Filali, and Christian Bonnet. Vehicular mobility simulation for vanets. In *Proceedings of the 40th Annual Simulation Symposium*, pages 301–309, Washington, DC, USA, 2007. IEEE Computer Society.

[For11] Ford Motor Company. Ford's intelligent vehicles, 2011. [Online; accessed October-2011].

[FP05] E. Ferro and F. Potorti. Bluetooth and wi-fi wireless protocols: a survey and a comparison. *Wireless Communications, IEEE*, 12(1):12 – 26, feb. 2005.

[FyLBS+06] Alaeddine El Fawal, Jean yves Le Boudec, Kave Salamatian, A. Self, and Limiting Epidemic Service. Self-limiting epidemic forwarding. Technical report, In the First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications, 2006.

[GBT+09] Dominic Greenwood, Branislav Burdiliak, Ivan Trencansky, Hartmut Armbruster, and Christian Dannegger. Greenwave distributed traffic intersection control. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1413–1414, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[GC08] B.S. Gukhool and S. Cherkaoui. Ieee 802.11p modeling in ns-2. In *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, pages 622 –626, oct. 2008.

[GE68] D.C. Gazis and L.C. Edie. Traffic flow theory. *Proceedings of the IEEE*, 56(4):458 – 471, april 1968.

[GK06] A. Giridhar and P.R. Kumar. Scheduling automated traffic on a network of roads. *Vehicular Technology, IEEE Transactions on*, 55(5):1467 –1474, sep. 2006.

[GM57] H.H. Goode and R.E. Machol. *System engineering: an introduction to the design of large-scale systems*. McGraw-Hill series in control systems engineering. McGraw-Hill, 1957.

[Gol10] Jennifer Golbeck. *Computing with Social Trust*. Springer Publishing Company, Incorporated, 2010.

[Goo07] Michael F. Goodchild. Citizens as voluntary sensors: spatial data infrastructure in the world of web 2.0. *International Journal of Spatial Data Infrastructures Research*, pages 24–32, 2007.

[Goo11] Google. Google maps. `http://maps.google.com`, October 2011.

[GPR+10] G.P. Grau, D. Pusceddu, S. Rea, O. Brickley, M. Koubek, and D. Pesch. Vehicle-2-vehicle communication channel evaluation using the cvis platform. In *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, pages 449 –453, july 2010.

[Hai63] Frank A. Haight. *Mathematical theories of traffic flow / [by] Frank A. Haight*. Academic Press, New York :, 1963.

[HBZ+06] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *In 4th ACM SenSys*, pages 125–138, 2006.

[Hel01] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73:1067, 2001.

[HFB09] J. Harri, F. Filali, and C. Bonnet. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *Communications Surveys Tutorials, IEEE*, 11(4):19 –41, 2009.

[HFBF06] J. Harri, F. Filali, C. Bonnet, and Marco Fiore. Vanetmobisim: generating realistic mobility patterns for vanets. In *Proceedings of the 3rd international*

workshop on Vehicular ad hoc networks, VANET '06, pages 96–97, New York, NY, USA, 2006. ACM.

[HFI+07] M. Hayashi, S. Fukuzawa, H. Ichikawa, T. Kawato, J. Yamada, T. Tsuboi, S. Matsui, and T. Maruyama. Development of vehicular communication (wave) system for safety applications. In *Telecommunications, 2007. ITST '07. 7th International Conference on ITS*, pages 1 –5, june 2007.

[HJ08] Ólafur Ragnar Helgason and Kristján Valur Jónsson. Opportunistic networking in omnet++. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems and workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[HKH07] Visit Hirankitti, Jaturapith Krohkaew, and Christopher J. Hogger. A multi-agent approach for intelligent traffic-light control. In *World Congress on Engineering*, pages 116–121, 2007.

[HKHL10] Kai-Yun Ho, Po-Chun Kang, Chung-Hsien Hsu, and Ching-Hai Lin. Implementation of wave/dsrc devices for vehicular communications. In *Computer Communication Control and Automation (3CA), 2010 International Symposium on*, volume 2, pages 522 –525, may 2010.

[HLP11] I.W.-H. Ho, K.K. Leung, and J.W. Polak. Stochastic model and connectivity dynamics for vanets in signalized road systems. *Networking, IEEE/ACM Transactions on*, 19(1):195 –208, feb. 2011.

[HRBW81] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton. SCOOT: A traffic responsive method of coordinating signals. Technical report, TRRL, 1981.

[HWH+10] Juan C. Herrera, Daniel B. Work, Ryan Herring, Xuegang (Jeff) Ban, Quinn Jacobson, and Alexandre M. Bayen. Evaluation of traffic data obtained via

gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18(4):568 – 583, 2010.

[IEE10] IEEE. IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services. *IEEE Std 1609.3-2010 (Revision of IEEE Std 1609.3-2007)*, pages 1 –144, 30 2010.

[Ino76] H. Inose. Road-traffic control with the particular reference to tokyo traffic control and surveillance system. *Proceedings of the IEEE*, 64(7):1028 – 1039, july 1976.

[Int08] Internet Society. Internet protocol, version 6 (ipv6) specification, 2008. [Online; accessed October-2011].

[JD08] D. Jiang and L. Delgrossi. Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036 –2040, may 2008.

[Jen08] E.J. Jenkins. *To scale: one hundred urban plans*. Routledge, 2008.

[JHP+03] Jorjeta G. Jetcheva, Yih-Chun Hu, Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. Crawdad data set rice/ad-hoc-city (v. 2003-09-11). Downloaded from http://crawdad.cs.dartmouth.edu/rice/ad-hoc-city, sep 2003.

[Kaw09] Nobuo Kawaguchi. Wifi location information system for both indoors and outdoors. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, IWANN '09, pages 638–645. Springer-Verlag, Berlin, Heidelberg, 2009.

[KCR+10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *Proceedings of the 2010 IEEE Symposium on*

Security and Privacy, SP '10, pages 447–462, Washington, DC, USA, 2010. IEEE Computer Society.

[KH95] Douglas A. Kurtze and Daniel C. Hong. Traffic jams, granular flow, and soliton selection. *Phys. Rev. E*, 52(1):218–221, Jul 1995.

[Kim07] Hyoungsoo Kim. *A Simulation Framework for Traffic Information Dissemination in Ubiquitous Vehicular Ad hoc Networks*. PhD thesis, University of Maryland, 2007.

[KOK09] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *SIMUTools '09: Proceeding of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ACM.

[Kom10] Paul Kompfner. Cooperative urban mobility handbook, 2010.

[KPM11] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Rose Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 127–140, New York, NY, USA, 2011. ACM.

[Kri08] Rajesh Krishnan. *Travel time estimation and forecasting on urban roads*. PhD thesis, Imperial College London, 2008.

[KS07] Frank Kargl and Elmar Schoch. Simulation of manets: a qualitative comparison between jist/swans and ns-2. In *Proceedings of the 1st international workshop on System evaluation for mobile platforms*, MobiEval '07, pages 41–46, New York, NY, USA, 2007. ACM.

[KSB09] Jonghyun Kim, Vinay Sridhara, and Stephan Bohacek. Realistic mobility simulation of urban mesh networks. *Ad Hoc Networks*, 7(2):411 – 430, 2009.

[LCM09a] Ilias Leontiadis, Paolo Costa, and Cecilia Mascolo. A hybrid approach for content-based publish/subscribe in vehicular networks. *Pervasive Mob. Comput.*, 5:697–713, December 2009.

[LCM09b] Ilias Leontiadis, Paolo Costa, and Cecilia Mascolo. Persistent content-based information dissemination in hybrid vehicular networks. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–10, Washington, DC, USA, 2009. IEEE Computer Society.

[Leo09] Ilias Leontiadis. *A Content Dissemination Framework for Vehicular Networking*. PhD thesis, University College London, 2009.

[Lew74] G.H. Lewes. *Problems of life and mind*. Number v. 1 in Problems of Life and Mind. Trubner & co., 1874.

[LH08] Stefan Lammer and Dirk Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics-theory and Experiment*, 2008.

[LHT+03] C. Lochert, H. Hartenstein, J. Tian, H. Fussler, D. Hermann, and M. Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 156 – 161, june 2003.

[Lit09] Todd Litman. Transportation cost and benefit analysis techniques, estimates and implications, January 2009.

[LJC+00] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130, New York, NY, USA, 2000. ACM.

[LLK99] Jee-Hyong Lee and Hyung Lee-Kwang. Distributed and cooperative fuzzy controllers for traffic intersections group. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 29(2):263–271, 1999.

[LM07] Ilias Leontiadis and Cecilia Mascolo. GeOpps: Opportunistic Geographical Routing for Vehicular Networks. In *Proceedings of the IEEE Workshop*

on Autonomic and Opportunistic Communications. (Colocated with WOWMOM07), Helsinki, Finland, June 2007. IEEE Press.

[LMP+07] Massimiliano Lenardi, Cornelius Menig, Timo Peichl, Matthias Rckl, Dieter Seeberger, Markus Straberger, Hannes Stratil, Hans-Jrg Vgel, Benjamin Weyl, and Wenhui Zhang. Car-2-Car Communication Consortium - Manifesto. *DLR Electronic Library*, 2007.

[Low82] P.R. Lowrie. Scats: Sydney coordinated adaptive traffic system - a traffic responsive method of controlling urban traffic, 1982.

[LW07] Fan Li and Yu Wang. Routing in vehicular ad hoc networks: A survey. *Vehicular Technology Magazine, IEEE*, 2(2):12 –22, june 2007.

[Mah07] Ratul Mahajan. CRAWDAD trace set microsoft/-vanlan/connectivity (v. 2007-09-14). Downloaded from http://crawdad.cs.dartmouth.edu/microsoft/vanlan/connectivity, September 2007.

[Mas06] Massachusetts Department of Transportation. Design build procurement guide. `http://www.mhd.state.ma.us/downloads/designGuide/CH_6_a.pdf`, January 2006.

[MBD10] Leonardo Mostarda, Rudi Ball, and Naranker Dulay. Distributed Fault Tolerant Controllers. In *10th IFIP international conference on Distributed Applications and Interoperable Systems (DAIS)*, Lecture Notes in Computer Science, May 2010.

[MGL04] Prasant Mohapatra, Chao Gui, and Jian Li. Group communications in mobile ad hoc networks. *Computer*, 37:52–59, 2004.

[MI04] Daniel Mahrenholz and Svilen Ivanov. Real-time network emulation with ns-2. In *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 29–36, Washington, DC, USA, 2004. IEEE Computer Society.

[MK94] Sadayoshi Mikami and Yukinori Kakazu. Genetic reinforcement learning for cooperative traffic signal control. In *International Conference on Evolutionary Computation*, pages 223–228, 1994.

[MKDL+11] M. Martino, K. Kloeckl, G. Di Lorenzo, J. Dunnam, E.R. Kang, and C Ratti. syn(c)ity: Vizualising the potential of a predictive in-car recommendation system. `http://senseable.mit.edu/papers/pdf/2010_Martino_et_al_Syn(c)ity_Internet_of_things.pdf`, December 2011.

[MM09] M. Musolesi and C. Mascolo. Car: Context-aware adaptive routing for delay-tolerant mobile networks. *Mobile Computing, IEEE Transactions on*, 8(2):246–260, Feb. 2009.

[MNBJ11] Bharati Mishra, Priyadarshini Nayak, Subhashree Behera, and Debasish Jena. Security in vehicular adhoc networks: a survey. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, pages 590–595, New York, NY, USA, 2011. ACM.

[MPR08] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 323–336, New York, NY, USA, 2008. ACM.

[MWH01] Martin Mauve, Jrg Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network*, 15:30–39, 2001.

[MWS+05] Rahul Mangharam, Daniel S. Weller, Daniel D. Stancil, Ragunathan Rajkumar, and Jayendra S. Parikh. Groovesim: a topography-accurate simulator for geographic routing in vehicular networks. In *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, VANET '05, pages 59–68, New York, NY, USA, 2005. ACM.

[NBB99] Kai Nagel, Richard L. Beckman, and Christopher L. Barrett. Transims for transportation planning. In *In 6th Int. Conf. on Computers in Urban*

Planning and Urban Management. Addison-Wesley, Reading,Massachusetts, 1999.

[NBG06a] Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 108–119, New York, NY, USA, 2006. ACM.

[NBG06b] Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 108–119, New York, NY, USA, 2006. ACM.

[NBH+11] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris. Smarter cities and their innovation challenges. *Computer*, 44(6):32 –39, june 2011.

[NDM+89] R.E. Neilan, T.H. Dixon, T.K. Meehan, W.G. Melbourne, J.A. Scheid, J.N. Kellogg, and J.L. Stowell. Operational aspects of casa uno '88-the first large scale international gps geodetic network. *Instrumentation and Measurement, IEEE Transactions on*, 38(2):648 –651, apr 1989.

[Ope11] Open Street Maps. Open Street Maps. `http://www.osm.org`, October 2011.

[OZRM00] L.E. Owen, Yunlong Zhang, Lei Rao, and G. McHale. Traffic flow simulation using corsim. In *Simulation Conference Proceedings, 2000. Winter*, volume 2, pages 1143 –1147 vol.2, 2000.

[Par09] Parliamentary Office of Science and Technology. Intelligent Transport Systems. http://www.parliament.uk/documents/post/postpn322.pdf, January 2009. [Online; accessed October-2011].

[PDAV08] H Van Dyke Parunak, Ph D, Ann Arbor, and Raymond S Vanderbok.

Managing emergent behavior in distributed control systems 1. *Control*, 1001(2007):1–8, 2008.

[PGHC99] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pages 1538 –1542 vol.3, 1999.

[PH09] S.W. Peters and R.W. Heath. The future of wimax: Multihop relaying with ieee 802.16j. *Communications Magazine, IEEE*, 47(1):104 –111, january 2009.

[PHL+10] Santi Phithakkitnukoon, Teerayut Horanont, Giusy Di Lorenzo, Ryosuke Shibasaki, and Carlo Ratti. Activity-aware map: Identifying human daily activity pattern using mobile phone data. In *HBU*, pages 14–25, 2010.

[PJP09] Giovanni Petri, Henrik Jeldtoft Jensen, and John W Polak. Global and local information in traffic congestion. *Europhysics Letters*, 88(2):4, 2009.

[PRL+08] M. Piórkowski, M. Raya, A. Lezama Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux. Trans: realistic joint traffic and network simulator for vanets. *SIGMOBILE Mob. Comput. Commun. Rev.*, 12:31–33, January 2008.

[PS96] B.W. Parkinson and J.J. Spilker. *Global positioning system: theory and applications*. Number v. 1; v. 163 in Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 1996.

[QWDFZ11] Bo Qin, Qianhong Wu, Josep Domingo-Ferrer, and Lei Zhang. Preserving security and privacy in large-scale vanets. In *ICICS*, pages 121–135, 2011.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM.

[Rob05] Steve Robinson. *The development and application of an urban link travel*

time model using data derived from inductive loop detectors. PhD thesis, Imperial College London, 2005.

[Rog07] R.M. Rogers. *Applied mathematics in integrated navigation systems*. AIAA education series. American Institute of Aeronautics and Astronautics, 2007.

[Ros85] Robert Rosen. *Anticipatory Systems: Philosophical, Mathematical and Methodological Foundations*. Pergamon, 1985.

[RT11] Carlo Ratti and Anthony Townsend. Harnessing Residents' Electronic Devices Will Yield Truly Smart Cities. *Scientific American*, August 2011.

[Sch88] Herb Schwetman. Using csim to model complex systems. In *Proceedings of the 20th conference on Winter simulation*, WSC '88, pages 246–253, New York, NY, USA, 1988. ACM.

[SHCD06] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A Networking Architecture Designed Around Mobile Users. *IFIP Conference on Wireless On demand Network Systems (WONS 2006)*, 2006.

[Sie11] Kurt Sievers. Nxp demonstrates new car-to-x communication platform, 2011. [Online; accessed October-2011].

[SKY99] Teo Lian Seng, Marzuki Khalid, and Rubiyah Yusof. Tuning of a neuro-fuzzy controller by genetic algorithms with an application to a coupled-tank liquid-level control system. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 29:226–236, 1999.

[SMBI10] M. Slavik, I. Mahgoub, A. Badi, and M. Ilyas. Design and implementation of parallel jist to support distributed wireless network simulation. In *High-Capacity Optical Networks and Enabling Technologies (HONET), 2010*, pages 154 –160, dec. 2010.

[SMR05] I. Stepanov, P.J. Marron, and K. Rothermel. Mobility modeling of outdoor scenarios for manets. In *Simulation Symposium, 2005. Proceedings. 38th Annual*, pages 312 – 322, april 2005.

[SN09] T. Sukuvaara and P. Nurmi. Wireless traffic service platform for combined vehicle-to-vehicle and vehicle-to-infrastructure communications. *Wireless Communications, IEEE*, 16(6):54 –61, december 2009.

[SPR05] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, New York, NY, USA, 2005. ACM.

[SRB+10] Enrico Scalavino, Giovanni Russello, Rudi Ball, Vaibhav Gowadia, and Emil C. Lupu. An opportunistic authority evaluation scheme for data security in crisis management scenarios. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 157–168, New York, NY, USA, 2010. ACM.

[SSV08] Peter Sanders, Dominik Schultes, and Christian Vetter. Mobile route planning. In *Proceedings of the 16th Annual European symposium on Algorithms*, ESA '08, pages 732–743, Berlin, Heidelberg, 2008. Springer-Verlag.

[Ste90] Luc Steels. Towards a theory of emergent functionality. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 451–461, Cambridge, MA, USA, 1990. MIT Press.

[Sus05] J.M. Sussman. *Perspectives on intelligent transportation systems (ITS).* Springer Science+Business Media, 2005.

[SV00] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383, 2000. 10.1023/A:1008942012299.

[TFD98] H. Taale, W.C.M. Fransen, and J. Dibbits. The second assessment of the scoot system in nijmegen. In *Road Transport Information and Control, 1998. 9th International Conference on (Conf. Publ. No. 454)*, pages 109 – 113, April 1998.

[THB+02] Jing Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Simulation Symposium, 2002. Proceedings. 35th Annual*, pages 337 – 344, april 2002.

[Thi11] Kyle Thibaut. Ford's V2V System Helps Driver Avoid Collisions. `http://translogic.aolautos.com/2011/06/22/fords-v2v-system-helps-driver-avoid-collisions-video`, 2011.

[Tho04] C. Thompson. Everything is alive. *Internet Computing, IEEE*, 8(1):83 – 86, jan-feb 2004.

[Thr10] Sebastian Thrun. What we're driving at. http://googleblog.blogspot.com/2010/10/what-were-driving-at.html, 2010. [Online; accessed October-2011].

[TIG04] TIGER/Line. Us geological survey (usgs) topographic maps, 2004.

[Tru04] S.A. True. Planning the future of the world geodetic system 1984. In *Position Location and Navigation Symposium, 2004. PLANS 2004*, pages 639 – 648, april 2004.

[TvA01] C. Tampere and B. van Arem. Traffic flow theory and its applications in automated vehicle control: a review. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 391 –397, 2001.

[TVB09] O.K. Tonguz, W. Viriyasitavat, and Fan Bai. Modeling urban traffic: A cellular automata approach. *Communications Magazine, IEEE*, 47(5):142 –150, may 2009.

[Var93] Pravin Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38:195–207, 1993.

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks, 2000.

[Vic00] R. Vickerman. Evaluation methodologies for transport projects in the united kingdom. *Transport Policy*, 7(1):7 – 16, 2000.

[Vin75] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, XXII, April 1975.

[VVPV97] H. Van, Raymond S. VanderBok, H. Van Dyke Parunak, and Ph. D. Raymond S. V. Managing emergent behavior in distributed control systems, 1997.

[Waz11] Waze. Real-time maps and traffic information based on the wisdom of the crowd. http://www.waze.com, 2011. [Online; accessed October-2011].

[WBT+10] Daniel B. Work, Sbastien Blandin, Olli-Pekka Tossavainen, Benedetto Piccoli, and Alexandre M. Bayen. A traffic model for velocity data assimilation. *Applied Mathematics Research eXpress*, 2010(1):1–35, 2010.

[Wei99] G. Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. Intelligent Robotics and Autonomous Agents. MIT Press, 1999.

[WER+03] L. Wischoff, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis - a self-organizing traffic information system. In *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, volume 4, pages 2442 – 2446 vol.4, april 2003.

[YLL+10] Qing Yang, Alvin Lim, Shuang Li, Jian Fang, and Prathima Agrawal. Acar: Adaptive connectivity aware routing for vehicular ad hoc networks in city scenarios. *Mob. Netw. Appl.*, 15:36–60, February 2010.

[YLN03] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312 – 1321 vol.2, march-3 april 2003.

[Zan09] Paul A Zandbergen. Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning. *Transactions in GIS*, 13:5–25, 2009.

A. Framework

The following appendix provides extra detail about the structure and operation of the Geographic Urban Simulator (GUS).

A.1. GUS Architecture and Execution

The Geographic Urban Simulator is an extension for the JiST platform. The combined JiST-GUS simulator comprises 104,301 lines of code, where the GUS extension makes up 32.4% of the simulator. Figure A.1 shows the object hierarchy. Benchmark simulations supported vehicle populations of up to 2218 vehicles per square kilometer. For comparison, the city of San Francisco typically deals with populations of about 3800 vehicles per square kilometer on a busy day. Due to resource constraints GUS has not been run with larger vehicle populations. Simulating large vehicle populations requires days of simulation. The GUS does not support parallel simulation, as JiST does not support it. However, work by Slavik et al. [SMBI10], has sought to enhance JiST for parallel computing, where clusters could be used to simulate a geographic region of a more extended road network. All experiments within the thesis were conducted on an Intel Core 2 Duo T9600 chip-set with two 2.8 GHz processors, supporting 3.72 GB of usable RAM. The GUS was run on Windows 7 64-bit. Simulations took place over hours, depending on the vehicle population size. The JVM typically used a maximum of 283 MB of memory during each simulation.

A single GUS simulation generates gigabytes worth of data depending on the length of a simulation scenario. The default simulation length was 3600 seconds (60 minutes of simulation time), where the GUS cycled every 100 milliseconds. This results in 36000 cycles per simulation of a single vehicle and about 36 million protocol cycles for a community of 1000 vehicles. The GUS recorded measurements for each messaging contact and mobility data, such that trace

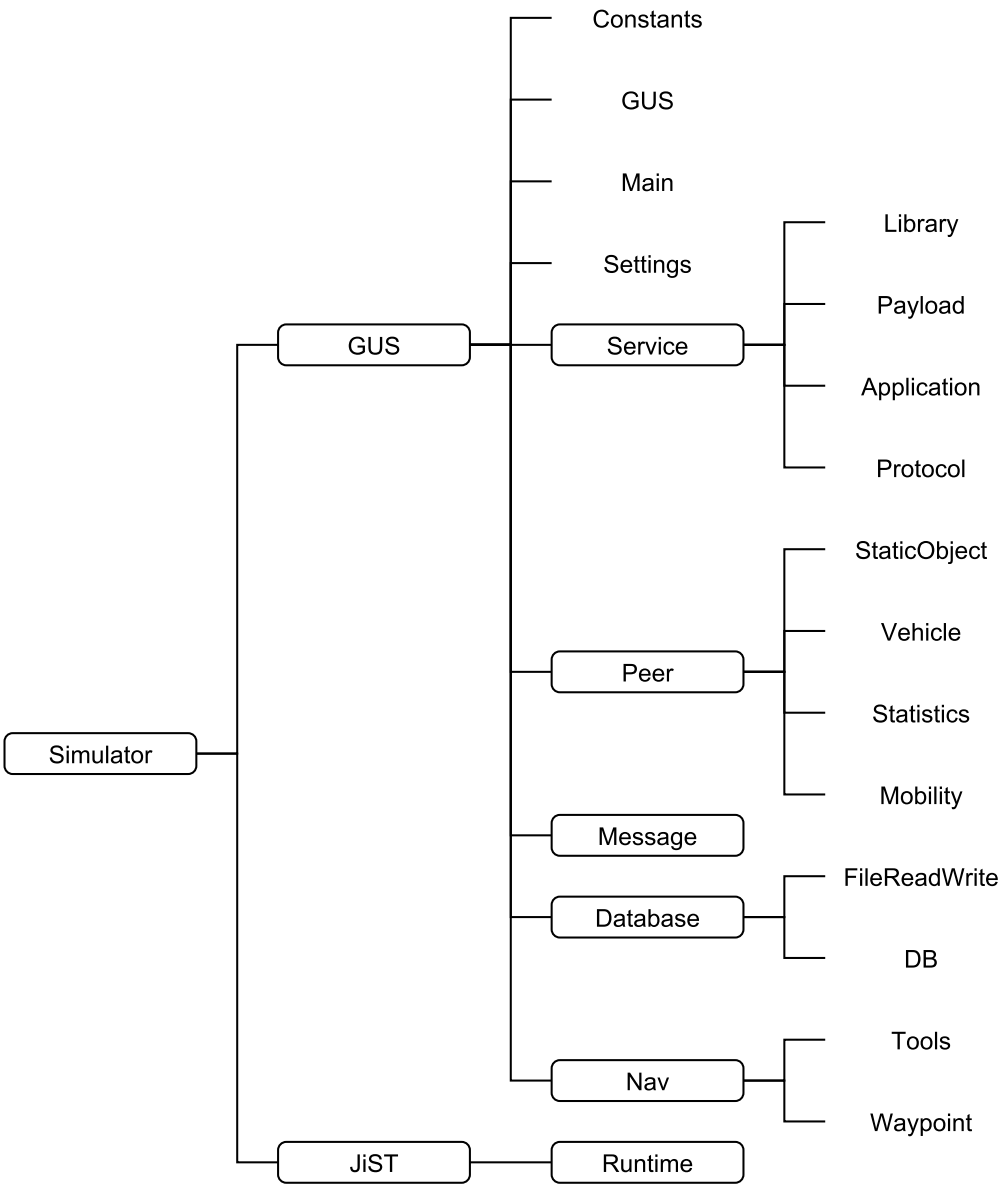


Figure A.1.: Geographic Urban Simulator (GUS) object architecture.

analysis and visualisation could occur post simulation. A number of helping tools are used to interpret data such as messaging interactions, dependencies and mobility. To handle these large amounts of data the GUS uses a MySQL database to store data. The database was also used to enable the simulation of larger vehicle populations, where memory limits where exceeded. The trade-off of using the database as an intermediary message store was such that simulation performance deteriorated. Messages which were normally cached in memory were now cached on disk and thus the read-write disk accesses increased as well as slowed the simulation process. Simulation parameters were provided as keyword command-line inputs or files. For instance, mobility traces provided using file inputs (for example London-traces.txt).

Listing A.1: Simulation parameters.

```
1 java jist.runtime.Main -Xms128m -Xmx512m simulator.Main
2       end=3600000
3       numpeers=200
4       commrange=250
5       mobility=London-traces.txt
6       broadcastinterval=5000
7       dropRecM=50
```

In the command-line example provided (Listing A.1), `end` specifies the duration of the simulation, `numpeers` the vehicle population to maintain within the scenario, `commrange` the maximum broadcast range for each message, `broadcastinterval` the interval between broadcasts (specified in milliseconds) and `dropRecM` the percentage of messages to drop during simulation. The GUS allows flexible miscellaneous keywords use such that keywords are service specific. In other words, a service developer can include new parameters which are not static keywords for providing protocol or application specific inputs. The protocols considered within the thesis, are in code an average of 126 lines of Java.

```
1 MessageQueue templateProtocol(MessageQueue inbox, long t, Waypoint p, Store s){
2       //Filtering
3       ...
4       //Processing
5       ...
```

```
6       //Finalisation
7       if (condition(t,p,s) == 0){
8             Payload p = new Payload();
9             Message msg = new Message(p);
10            ...
11            outbox.push(msg);
12      }
13      return outbox;
14}
```

The GUS does not impede development of data-structures, which may be used by a service protocol or application. An example of such an included and in-line data-structure is the MapStore (Section B.1). All code developed in the framework is directly transferable for Android Dalvik[1] compilation and use as JiST limits its support to Java 1.4.2.

A.2. Trace Synthesis

GUS mobility traces were sourced from both a scraping scripts and from datasets used by related work, including the ETH trace datasets [NBG06a]. Synthetic mobility traces were constructed using GPX and KML driving directions [2], which themselves also contain other road meta-data (Listing A.2) from Google[3] and the Open Street Map[4] databases.

Listing A.2: Code snippet to query GPX driving directions between A and B.

```
1 def getGPXDirections(A,B):
2       url = "http://navigation.cloudmade.com/45d5e7249f5f478ba704ac5202372006/" +
3             api/0.3/" + A + "," + B + "/car.gpx?tId=CloudMade"
4       gpx = urllib.urlopen(url).read()
5       return gpx
```

Extracted GPX directions are filtered to remove the driving directions as a list of waypoints. The gaps between waypoints are filled or expanded to improve the granularity of the traces. In other words, the algorithm increases the resolution of the trace, making mobility more fine grained. The final result of synthesis is a specified trace of the form in Listing A.3, where inter-

[1] http://code.google.com/p/dalvik
[2] http://www.cloudmade.com
[3] http://maps.google.com
[4] http://www.openstreetmap.org/

mediary waypoints (geographic positions) are delimited by colons. In the mobility specification, we specify the type of mobility trace for identification and the in-order mobility waypoints of the vehicle. Notably, the maximum speed limit is not specified.

Listing A.3: A short synthetic mobility specification.

```
type=London-Individual-Trace mobility=[(52.538902,13.381104):(52.538734,13.381275):
(52.538326,13.381703):(52.537704,13.382364):(52.537113,13.38306):(52.536701,13.383546):
(52.535995,13.384395):(52.535931,13.384471):(52.535191,13.385353):(52.534912,13.385669):
(52.53466,13.385943):(52.534306,13.386357):(52.533852,13.386961):(52.533508,13.387395):
(52.53347,13.387447):(52.533382,13.387576):(52.533348,13.387625):(52.533321,13.387667):
(52.533035,13.388051):(52.531734,13.389748):(52.531868,13.390995):(52.532047,13.392756):
(52.532143,13.393675):(52.532181,13.394051):(52.532242,13.394501):(52.532299,13.395001):
(52.532349,13.395454):(52.532387,13.395815):(52.53241,13.395963):(52.53244,13.397037):
(52.532436,13.397168):(52.532459,13.397774):(52.532471,13.398007):(52.532497,13.398618):
(52.532524,13.398791):(52.532578,13.398952):(52.532501,13.399004):(52.53215,13.399263):
(52.531902,13.399449):(52.531704,13.399616):(52.531307,13.399939):(52.530334,13.400775):
(52.530033,13.401033):(52.529911,13.401068):(52.529739,13.40119):(52.529617,13.401269):
(52.529625,13.4015):(52.529652,13.402007):(52.529724,13.403261):(52.529713,13.403615):
(52.529678,13.403888):(52.5294,13.40548):(52.528755,13.409147):(52.528709,13.409405):
(52.528656,13.409641):(52.528603,13.40992):(52.528378,13.411175):(52.52816,13.411959)]
```

A.3. Geographic Distance

Geographic positioning is, always provided with an accuracy error. This error, may be a few centimetres or hundreds of meters, depending on the quality of the data available from GPS signals and the calculations made to trilaterate position. Assuming that positions are correct, geodesic distance calculations determine the distance of one position from another. Adding to the complexity is that the Earth is not a perfect sphere, but rather a geoid. Within cities, these structures are most commonly man-made, however a large number of structures which are primarily responsible for man-made structures are natural (for example hills, mountains, rivers, lakes). The most common method of calculating distance is the usage of the Haversine formula, however the formula and law of cosines are still in some cases inaccurate. A counter to this is that we use the more modern and precise Vincenty algorithm (Listing A.4) in com-

bination with the WGS84 standard to calculate distance, to accuracies (on average) within millimetres.

Listing A.4: The Vincenty algorithm [Vin75] in Java.

```
public double calcVincentyDistance(Waypoint posA, Waypoint posB){
      double lat1 = posA.getLatitude(), lon1 = posA.getLongitude();
      double lat2 = posB.getLatitude(), lon2 = posB.getLongitude();
      double a = 6378137, b = 6356752.3142, f = 1/298.257223563; // WGS-84 ellipsoid
      double L = this.degreesToRadians(lon2-lon1);
      double U1 = Math.atan((1-f) * Math.tan(this.degreesToRadians(lat1)));
      double U2 = Math.atan((1-f) * Math.tan(this.degreesToRadians(lat2)));
      double sinU1 = Math.sin(U1), cosU1 = Math.cos(U1);
      double sinU2 = Math.sin(U2), cosU2 = Math.cos(U2);
      double lambda = L, lambdaP, iterLimit = 100, cosSqAlpha = 0, sinSigma = 0,
      double sinLambda = 0, cosSigma = 0, cosLambda = 0, sigma = 0, sinAlpha = 0;
      double cos2SigmaM = 0;

      do {
            sinLambda = Math.sin(lambda);
            cosLambda = Math.cos(lambda);
            sinSigma = Math.sqrt((cosU2*sinLambda) * (cosU2*sinLambda) +
            (cosU1*sinU2-sinU1*cosU2*cosLambda) *
            (cosU1*sinU2-sinU1*cosU2*cosLambda));

            if (sinSigma==0) return 0; // co-incident points
            cosSigma = sinU1*sinU2 + cosU1*cosU2*cosLambda;
            sigma = Math.atan2(sinSigma, cosSigma);
            sinAlpha = cosU1 * cosU2 * sinLambda / sinSigma;
            cosSqAlpha = 1 - sinAlpha*sinAlpha;
            cos2SigmaM = cosSigma - 2*sinU1*sinU2/cosSqAlpha;

            if (Double.isNaN(cos2SigmaM)){ cos2SigmaM = 0; }

            double C = f/16*cosSqAlpha*(4+f*(4-3*cosSqAlpha));
            lambdaP = lambda;
            lambda = L + (1-C) * f * sinAlpha * (sigma +
            C*sinSigma*(cos2SigmaM+C*cosSigma*(-1+2*cos2SigmaM*cos2SigmaM)));
```

```
34        }
35        while (Math.abs(lambda-lambdaP) $\textgreater$ 1e-12 && --iterLimit $\textgreater$ 0);
36
37        if (iterLimit==0){ return Double.NaN; }
38        else {
39              double uSq = cosSqAlpha * (a*a - b*b) / (b*b);
40              double A = 1 + uSq/16384*(4096+uSq*(-768+uSq*(320-175*uSq)));
41              double B = uSq/1024 * (256+uSq*(-128+uSq*(74-47*uSq)));
42              double deltaSigma = B*sinSigma*
43              (cos2SigmaM+B/4*(cosSigma*(-1+2*cos2SigmaM*cos2SigmaM)
44              - B/6*cos2SigmaM*(-3+4*sinSigma*sinSigma)*
45              (-3+4*cos2SigmaM*cos2SigmaM)));
46
47              return b*A*(sigma-deltaSigma);
48        }
49  }
```

A.4. Geographic Bearing

Where the course represents the directional travel of an object, the bearing provides the direction in which one object is from another. The forward azimuth is used to calculate the bearing between a source and destination position (Listing A.5). In the context of marine course and bearing calculations, the track describes the actual path of an object through space over time.

Listing A.5: Calculating precise bearing in Java.

```
1  public double getBearingBetween(Waypoint from, Waypoint to){
2        Waypoint posB = to;
3        Waypoint posA = from;
4        double result = 0.0;
5        double latA = posA.getLatitude();
6        double lonA = posA.getLongitude();
7        double latB = posB.getLatitude();
8        double lonB = posB.getLongitude();
9        latA = this.degreesToRadians(latA);
10       latB = this.degreesToRadians(latB);
```

```
11        double dLon = this.degreesToRadians(lonB - lonA);
12        double y = Math.sin(dLon) * Math.cos(latB);
13        double x = Math.cos(latA)*Math.sin(latB) - Math.sin(latA)*Math.cos(latB)*Math.cos(dLon);
14        result = Math.atan2(y, x);
15        result = result * (180/Math.PI);
16        result = (result + 360) % 360;
17        return result;
18    }
```

A.5. Library Function Dependencies

Functions within the function library are geodetic (Figure A.2). All functions are derived from the primitive position and time data collected by a vehicle over time. The hierarchy illustrates the inclusion of higher level functions in the construction of lower level functions. Typically functions further from the base primitives require more complex heuristics.

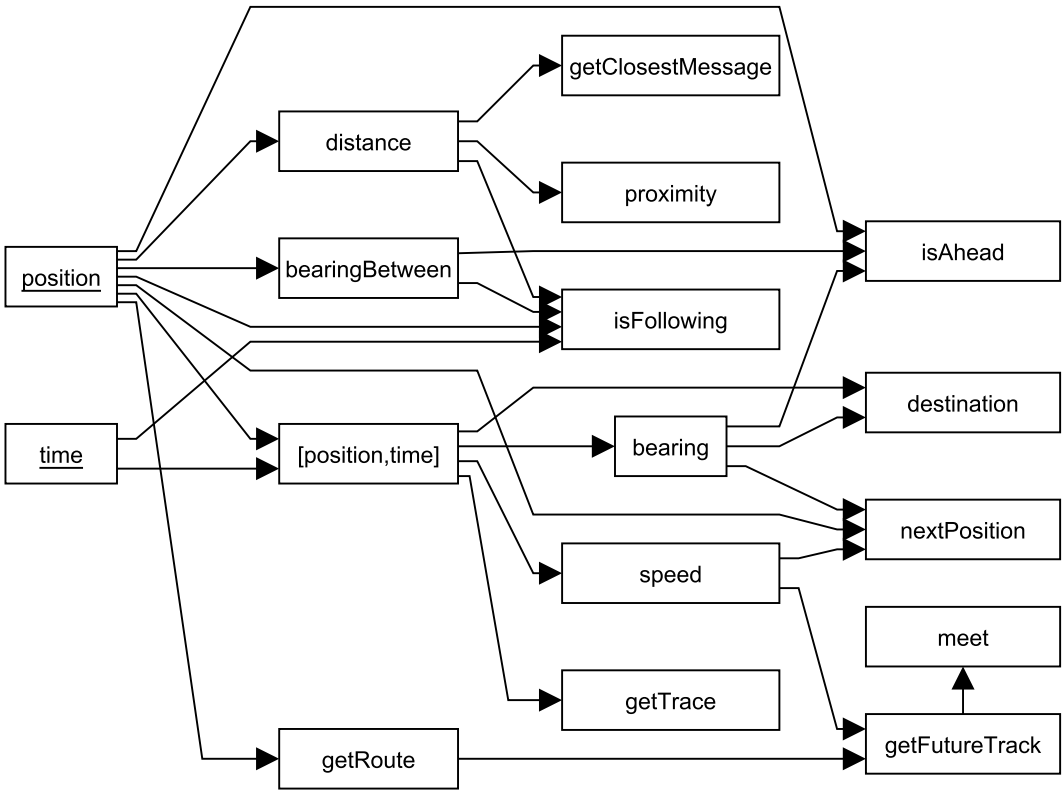


Figure A.2.: Hierarchy of derived library functions.

B. Travel Time Estimation

The following appendix contains extra information about the MapStore and its data-structures as well as graphs which visually present the raw individual growth of MapStore.

B.1. MapStore Datastructure

The MapStore is a travel time tuple storage data-structure (Listing B.1). It is however not optimised, as metrics were recorded and used to understand the behaviours of the travel time estimation service over time. MapStore operation provides an indication on how well the travel time estimation service performs at finding at least a single travel time for all road sections.

Listing B.1: MapStore methods.

```
1  class MapStore {
2        ArrayList<String> index;
3        HashMap<String,ArrayList<Tuple>> fragments;
4
5        int countAlreadySeen = 0;
6        int numberOfTuples = 0;
7        public int getNumberOfTuples()
8        public int getKeySize()
9        public int getValueSize()
10       public MapStore()
11       public void mergeTuple(Tuple t)
12       public ArrayList getTravelTimes(String section)
13       private void tallyEdgeRecorded(StringBuffer edge)
14       public boolean haveSection(String section)
15 }
```

Each MapStore stores unique travel time tuples, however we may have multiple travel times associated with a particular road section. The MapStore associates road sections with zero

or more travel times, each with unique timestamps. The MapStore index stores the unique identities of road sections (keys) with multiple travel times (values). In other words, a particular road section may have many sampled travel times associated with it. As the road network changes over time, a vehicle is capable of storing samples from different time periods. For example, a road section may have a different travel time at different times during the morning and evening. We assume that the service application component handles the interpretation of this travel time data. This also poses a problem for selection algorithms as a strategy must be chosen concerning which data to share given the bandwidth limitations of WAVE Short Messages (WSM). The counter `countAlreadySeen` is incremented each time a redundant tuple is merged by error. The `numberOfTuples` counter is updated each time a new Tuple is added to the MapStore successfully.

C. Intersection Control

The following appendix contains pseudo-code describing the intersection control collision algorithm, a description of the regional divisions made by a vehicle for the filtering of messages and the dependencies exhibited as a result of adaptations occurring between vehicles.

C.1. Collision Avoidance Algorithm

Algorithm 3 is used to estimate whether two or more vehicles are to collide in a future time instance. The algorithm determines and calculates the collision waypoint position (X) where two sets of future tracks cross. When vehicles are compared against one another, we are presented with one of three possible collision scenarios; either (a) collision while crossing, (b) collision while following or (c) no collision. The algorithm is provided the future tracks of two vehicles (future tracks A and B) and a specified minimum the separation distance (D). If no collision is expected to occur the algorithm does not return the estimated displacements to a collision point and a collision waypoint (X).

Future tracks conform geometrically to the road intersection. Where two waypoints are less than or equal to D we consider this position the collision position X. In simulations D was recalculated at each cycle to accommodate the changing speed of a vehicle. Where a collision is detected, the displacement distances U and V are returned as well to determine which vehicle is closer to X (distance calculations are not sufficient as the intersection contains turns). Future tracks are in effect compared 'side-by-side' and the absolute difference in displacement is compared to the D required for safe avoidance.

Algorithm 3: Collision Algorithm estimates the mobility of two vehicles (given their future tracks A and B) to determine whether a collision is likely to occur in a future time-step (t). The algorithm returns the displacement distances of each vehicle (U and V) and the collision point (X)

Input: `futuretrack` A, `futuretrack` B, `separation distance` D
Output: `distance` U, `distance` V, `Waypoint` X

```
1  begin
       // Assuming that length(A) = length(B)
       // Temporary collision flag
2      F ← false
       // For each future track position in time (t)
3      for t ← 1 to length(A) do
           // Update the calculated displacements of the future tracks
              A and B
4          U ← U + distance(A[t - 1], A[t])
5          V ← V + distance(B[t - 1], B[t])
           // Calculate the distances between the estimated future
              position waypoints
6          if distance(A[t], B[t]) ≤ D then
7              F ← true
8              X ← A[t]
9              break
10     return (U,V,X)
```

C.2. Vehicle Regions

Each vehicle divides its local space into long-range and short-range regions (Figure C.1). A region directly ahead of a vehicle is a considered of special importance for the avoidance of immediate obstacles which may threaten the vehicle with collision. The Vehicle Back-off Protocol (VBP) prioritises messages and sensory data which is located within this danger region and filters messages based on whether they exist within the short and long range regions. The vehicular networks constructed within experiments illustrate both the mean contact counts and the dependencies formed due to adaptations (A). Table C.1 considers the mean number of contacts per vehicle, given received messages from and, of that set, the mean number of vehicles with which contention existed and adaptation was required. As vehicle densities increased, increased contention exists between vehicles to use the intersection, however the maximum mean adaptations was limited to 6 interactions.

Figure C.1 depicts the regional space for a vehicle travelling in a vectored direction (left to right). As vehicles turn or change their bearing the regional overlay changes and therefore

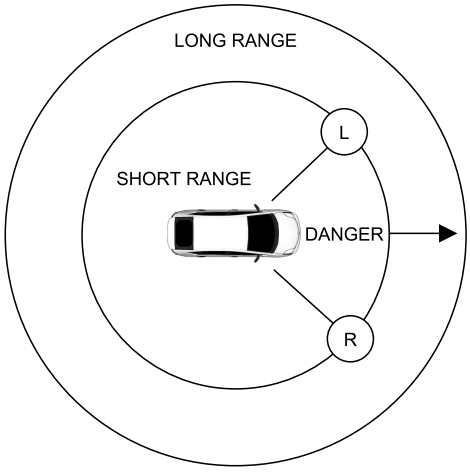


Figure C.1.: Vehicle regions. A moving vehicle maps short and long range locations to regions. A danger region directly ahead of the vehicle is defined between two left (L) and right (R) points. The danger sector can be made dynamic changing with the speed of a vehicle. Obstacles detected within the danger sector in front of a vehicle are handled with priority.

alters the sets of messages filtered.

C.3. Contacts and Dependencies

Input Rate	500	750	1000	1250	1500
Mean Contacts	10	16	22.5	31.5	43.5
Mean Adaptation	1	2	3.5	5	6
σ	1	1	2	3	3

Table C.1.: Mean number of contacts versus mean number of adaptations contacts (and associated standard deviation σ).

A common problem seen in road networks is the transmission of dependency between groups of vehicles. The actions of a vehicle in a previous time has the possibility of being transmitted over subsequent collections of vehicles. Such as is seen in an example dependency graph (Figure C.2). The graph illustrates the delay dependencies between interacting vehicles. The higher the edge weight between transmitting vehicles the larger the incurred delay to a vehicle by another vehicle. Vehicles with lower identifiers are older than those with higher identifiers.

An example dependency exists between 185 and 183. 185 adapts its behaviour to accommodate the mobility of 183. Linked vehicles have indirect effects on one another as well. For example, vehicle 172 was indirectly effected by 166, even though they never made contact

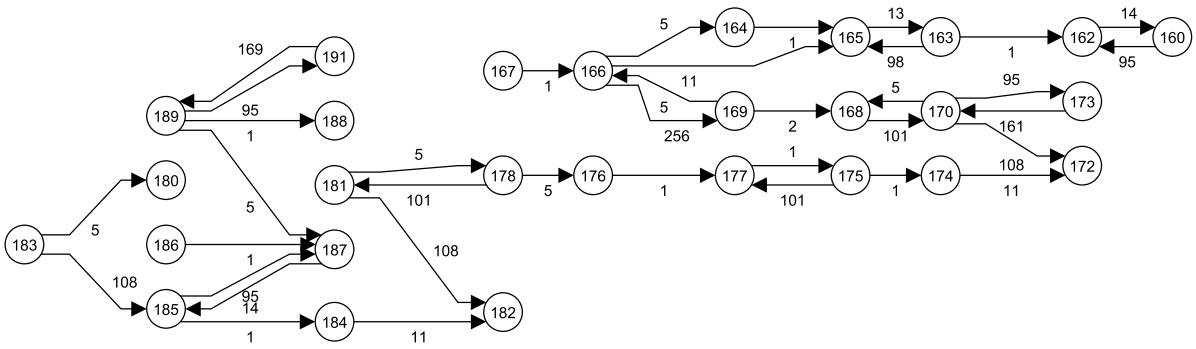


Figure C.2.: An example dependencies graph showing the linked adaptations between vehicles (nodes) as they negotiate their ordering to use the intersection. The dependency graph allows us to see which vehicles influenced the mobility of which other vehicles and hence provides a method of representation of run-time behaviour. Edge weights represent the number of adaptation negotiations occurring between two vehicles. The vehicle seen to receive more adaptations typically yields.

directly. We measure dependencies in terms of complete and separated adaptation graphs. Adaptation is the consequence of contention. A complete dependency graph suggests that all vehicles are through interconnections dependent on a previous vehicle from a previous time step. Disconnected dependency graphs are seen in scenarios for lower input rates. Dependency graphs are useful in understanding the behaviour of a road section, as they highlight vehicles and actors within the scenario which have the most influence on the operation of the system.