Final Year Project Report

Extended Version of SUMO for Real Time Vehicles' Routing in Smart Cities

David Smith

A Final Report submitted in part fulfilment of the degree of

BSc (hons) in Computer Science

Supervisor: Professor John Murphy

Mentor: Dr. Soufiene Djahel

UCD School of Computer Science and Informatics

College of Science

University College Dublin

March 11, 2014

Table of Contents

Project Specification

Subject: Road Traffic Simulation, Routing Algorithms
Project Type: Design and Implementation
Software Requirements: XML, Python
General Information: Road traffic congestion is a serious problem that most of large cities suffer from. We distinguish two types of congestion, recurrent and non recurrent. Recurrent congestion occurs usually when too many vehicles use the limited space of road network at the same time (e.g. weekday morning and afternoon peak hours). As for non-recurrent congestion, it mainly results from random events like traffic incidents (e.g. crash or stalled vehicles), work zones, bad weather conditions and some special events. In order to deal with traffic congestion, the traffic management authorities envisage applying load balancing techniques to prevent overloading some road segments in the cities. To this end, the traffic operator needs to get real time traffic flow information and be able to adapt (change) the shortest (fastest) route previously assigned to each car based on the traffic information update, congestion level and incidents.

The purpose of this project is to contribute to the ongoing efforts on updating a vehicles route during its travel from the departure location to the arrival point. To achieve this goal, the student will extend the microscopic road traffic simulator SUMO (Simulation of Urban MObility, http://sumo.sourceforge.net/) by developing additional features that allow us to update the vehicles shortest path during simulation runtime instead of using static predefined routes (current version of SUMO). In addition, the student will implement some heuristics for optimal route calculation (to be decided later) and extend them by incorporating new route selection metrics such as:

- Travel time (fastest route)
- Route with less fee (avoid tolls) and less fuel consumption
- Easiest route for driving (less number of turns, larger road lanes etc)
- Combination of two or more metrics

The student may also provide additional information related to the car characteristics (type, height) which will be used by the traffic operator to choose the adequate optimal route for this car.

Mandatory:
1- Familiarize with SUMO and TraCI
2-Design and implement a mechanism to update the vehicles' routes during simulation runtime.
Discretionary:
1-Implement some vehicular routing heuristics
2-Extend these heuristics by adding more route selection metrics:
- Travel time (fastest route)
- Route with less fee (avoid tolls) and less fuel consumption
- Easiest route for driving (less number of turns, larger road lanes etc)
- Combination of two or more metrics

Exceptional:
Use VANETs (Vehicular Ad hoc Networks) applications to control vehicles mobility during simulation runtime.

Abstract

The purpose of this project is to contribute to the ongoing efforts aiming to design solutions that enable updating vehicles' routes during simulation runtime. To achieve this goal, we will extend the microscopic road traffic simulator SUMO (Simulation of Urban MObility), by developing additional features that will allow the simulator to update the vehicles' defined route in case of an incident on the road, instead of using the statically defined routes. This report will describe the background research that has been done, including the current state of the art and related works proposed to improve SUMO efficiency and accuracy. Moreover, we will discuss some of its missing features, then we will explain in detail how to simulate an accident in SUMO and evaluate its impact on the commuters travel time and the overall traffic congestion level. Afterwards we will describe how our re-routing mechanism works along with the effect it has on improving travel time compared to the basic SUMO.

Acknowledgements

I would like to express my gratitude to Professor John Murphy, my project supervisor and Dr. Soufiene Djahel, my mentor, for their patience and guidance, their encouragement and useful critiques of my work. I would also like to thank Shen Wang who helped me without hesitation when I needed his assistance with TraCI and SUMO.

Chapter 1: **Introduction**

At the core of this project is the problem of traffic congestion and how it can be alleviated. In large cities, traffic congestion is caused by the volume of traffic closely approaching the maximum capacity of the road network. During rush hours it gets worse and with more people joining the road network every day, this problem will not disappear on its own. It is infeasible for a government to match a road network improvement programme to the unrestricted trends in traffic growth.

The economic loss due to traffic congestion is a huge factor, in the 39 metropolitan areas of the US with a population of 1 million or more, roughly one third of all vehicular travel occurs under congested conditions where the average speed is half of its free flow value. About half of the congested traffic is on express ways, causing a delay of over half a minute per kilometre of travel. The other half is on other arterial roads where the delay amounts to 1.2 minutes per kilometre of travel. With some 75 million licensed drivers in heavily populated areas, each averaging roughly 16,000 kilometres per year in those areas, there are 1.2 trillion kilometres driven in metropolitan areas, this amounts to a total delay of 6 billion hours [1]. Back here in Europe the economic cost is an even bigger factor, with estimates of 200 billion Euro in losses due to traffic congestion in 2012. This is roughly 2% of Europe's GDP, and is more than double the American estimate of $101 billion [2]. So it is clear from this data that there is a significant enough reason for traffic congestion to be alleviated, and efforts have been made by navigation system companies. But these systems are still in the early stages of becoming fully fledged and highly reliable products, therefore in order to develop a more sophisticated system simulation tools must be used to gain more insights on the traffic flow patterns on road networks and how this traffic flow evolves if an accident occurs or road works are taking place.

In Europe, a Dutch manufacturer called TomTom NV [5] has developed navigation systems for drivers and pedestrians, a US company Garmin [6] also developed a similar system. Both systems use GPS in order to establish the location and some routing algorithms to establish (generally) the fastest route to the destination. In some cases we can choose to avoid toll roads and motorways. TomTom has recently developed a system, known as TomTom Traffic, which provides users with precise and accurate real time information based on the state of the traffic and any congested routes ahead. It pinpoints exactly where the expected delays start and end. If the traffic situation changes then TomTom will continuously look for the fastest route. It puts the users in control, by informing them where incidents have occurred and giving them real time congestion information, so that they can choose whether to stay on course or take a detour. Garmin offers a similar live service that updates every two minutes to check traffic situations. Data is pulled from millions of other users, including mobile phone users, incident reports, radio feeds, news feeds, historical traffic data and fixed traffic sensors. These systems have only recently been introduced in Ireland, with TomTom Traffic only going live in 2011, and Garmin in 2010. Although these systems pull data from a lot of users, tens of thousands in Ireland, it is broadly limited by comparison to the amount of actual road users. It also uses the number plate registration systems on motorways. This is a fixed infrastructure installed by local authorities that tracks a number plate and how long it takes to get from one point to another. Again these systems are only installed on major motorways, and do not take into account arterial roads approaching or exiting these motorways. It is clear that although these systems are currently in use by millions of people around the world, their scope is rather limited when it comes to specific areas not monitored by sensors.

The main aim of this project is to develop a system using the microscopic traffic simulator (SUMO) that ensures real time update of the drivers routes (i.e. dynamic route update) upon detection of any abnormal increase in traffic congestion or as consequence of an accident. The end goal is to develop a model that will update a vehicles route when an accident occurs. The updated route will be based on the shortest path from the vehicles location (upon detection of an accident), to its destination, excluding the road where the accident occurs.

The rest of the report is organised as follows. In Chapter 2 we give an overview of the current state of the art in road traffic simulation, and the related works that pertain to this project, then we describe the missing features in SUMO. The impact of accidents on road traffic congestion is evaluated in Chapter 3, and the results from various scenarios are presented. Chapter 4 explains our proposed solution and implementation, and the results illustrating the performance of our mechanism are evaluated. Finally Chapter 5 concludes the report and describes the work that can be done in the future.

Chapter 2: **Background Research**

2.1 State of the Art

Inter vehicle communication (IVC) is currently transitioning from academic research to a feasible technology. However many aspects of IVC protocols, their parameters and configurations, as well as application specific adaptations require further research, a key aspect of this research in studying these IVC systems is simulation. In the past few years significant improvements have been made in the amount of realism that can be achieved using these simulators. In a paper by Stefan Joerer *et al* [4], the authors review the various simulation studies published at major vehicular conferences from 2009 to 2011, with a focus on reproducibility and comparability of these studies. The paper discusses the use of network simulators, physical layers and medium access, however we will just focus on the traffic simulator used. They have found that the use of certain mobility models have substantial influences on metrics like the number of unreachable nodes, the average path length and topology changes. In the paper they focus on the car-following model because most of the microscopic road traffic simulators are based on this class of model. This is because the car-following models derive their speeds based on the velocity and distance of the vehicle and those ahead of it. This is a realistic representation of how drivers would set the speed of their car. Whereas models inspired by cellular automations divide roads into sections of a certain length that can be either empty or completely occupied by one vehicle. The velocity of a vehicle is modelled by occupying multiple segments in one discrete time step. The Wiedemann model was the first car following model, it has now been further developed in order to take into account other aspects of drivers, such as their physical or psychological state. It is currently employed in the VISSIM traffic simulator [7]. The two other car following models are Gipps and Intelligent Driver Models, both of which are available in SUMO. As we can see from the Figure below, a lot of the papers they reviewed did not specify what simulator was used, but regarding the simulators that were specified SUMO was the most popular.

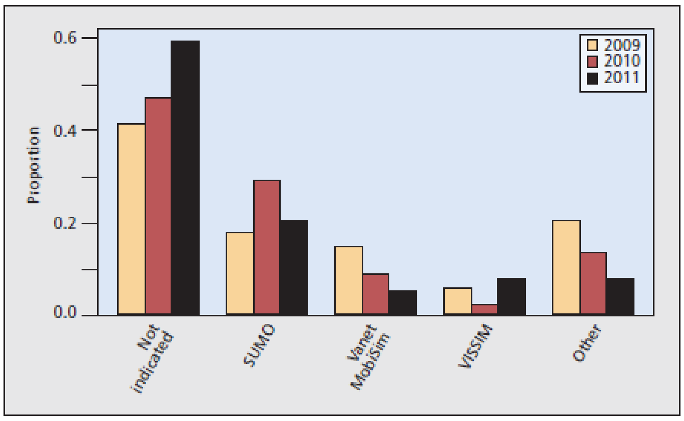

Figure 2.1: Distribution of road traffic simulators. [4]

PTV VISSIM is a microscopic multi-modal traffic simulator. It was developed by PTV in 1992 and is now the global market leader. Multi-modal simulation describes the ability to simulate more than one type of traffic, all these different types can interact mutually. In VISSIM these types are Vehicles, Public Transport, Cycles, Pedestrians and Rickshaws.

Unlike the open source traffic simulators, such as SUMO, VISSIM is a commercial simulator. It is a highly realistic simulator, but since the sources are not available, extensions to the model cannot be built, i.e. experimentation is limited. Other commercial simulators include PARAMICS [8] and AIMSUN [9]. The fact that these simulators have an expensive licensing scheme, which is a major impediment to the academic research community, they are not widely adopted by academics. They are more regularly used by government departments such as transport or emergency services.

VanetMobiSim [10] is an open-source extension to CanuMobiSim user mobility framework, it is capable of producing realistic vehicular mobility traces for several network simulators.

The Street Random Waypoint (STRAW) [11] tool is another mobility simulator, based on the freely available Scalable Wireless Ad Hoc Network Simulator (SWANS). STRAW is able to parse TIGER [12] files and it also implements complex intersection management, however due to its reliance on SWANS any research using STRAW on other network simulators is not permitted.

Simulation of **U**rban **MO**bility, (SUMO) [13], is an open source, microscopic, multi modal traffic simulator. It allows the user to simulate how a specified traffic demand performs on a given road network. It is microscopic, which means each vehicle is modelled explicitly, it has its own route and moves individually through the network. The German Aerospace Center began developing SUMO in 2001 and since then it has been improved and has evolved into a suite of traffic modelling utilities which includes a road network capable of reading different source formats, demand generation and routing utilities. SUMO was developed as an open source simulator in the hopes that people like me and others working with the simulator could suggest and implement improvements to the simulator helping to build a better and more realistic model. SUMO is not only a traffic simulator, but more so a suite of applications allowing the user to prepare a road network and traffic demand for this network. It uses "netconvert" to import a network from Open Street Map or from other traffic simulators such as VISUM, MATsim or VISSIM. Once we have imported and converted a road network it will need traffic demand, and routes for each vehicle. "*DUAROUTER*" is one of tools used to computing routes, "*DFROUTER*" is another. These routing tools take the network and trips as arguments and produce a route file that contains the routing information for each vehicle defined in the network. We can use random trips (which is what has been used in this project), or we can manually create a demand using OD (Origin-Destination) matrices or even by supplying various parameters for the specified network. These parameters include the population and the land class usage definition, among others.

In 2006, SUMO was extended by allowing it to interact with an external application via a socket connection. This was developed by Axel Wegner *et al* from University of Lubeck, and was incorporated into SUMO's official release, this implementation refers to TraCI [14]. TraCI is used as a device that can alter the state of the SUMO simulation during runtime.

In a research paper by Vi Tran Ngoc Nha *et al* [15], the various routing algorithms that could be used in conjunction with SUMO or another traffic simulator were described and deeply analysed based on their merits and limitations. The first of these algorithms that would be a suitable candidate of dynamic routing is Dijkstra. This algorithm finds the shortest path with the lowest cost from one node to all the nodes in a city map. Dijkstra is a worthwhile algorithm because it terminates once the destination node is found, i.e. shortest path found. Some other algorithms are unable to determine the shortest path until all the nodes are formed into an shortest path tree. An alternative to this would be A* which uses a heuristic function instead of an optimal search algorithm, A* is able to then restrict the search space which in turn improves computation time, the search space would be reduced to the area where an incident has occurred, such as an accident or sheer traffic volume. The authors then discuss how these route planning algorithms can be improved. For example the parameters that can be used as inputs for vehicles, such as road information, the current state of traffic and congestion on certain road segments, the destination information for the journey. Along with mobile information such as the fuel level, and other vehicle conditions and driver conditions, such as whether or not they are tired or need a break. The best route selection criteria and algorithm evaluation metrics are discussed, to identify the most appropriate route the metric's must be defined, such as travel distance,travel time, the ease of driving and the travel cost.

2.2 Related Works

INRIX, Inc.[16] is a provider of traffic information, it provides historical and real-time traffic information throughout the US and Canada and also most of Europe and Brazil. INRIX collects information about road speeds from almost 100 million anonymous mobile phones, delivery vehicles and lorries, along with other various fleet vehicles that have been equipped with a GPS device. All data collected is processed in real time and used to create traffic flow information for motorways and artirials across its user space.

As it was mentioned in the introduction, TomTom and Garmin have developed systems that alert drivers of any congestion ahead before they reach it. Both TomTom and Garmin have a partnership with INRIX as a part of their data collection process, they also use many other sources as was mentioned above. They are both commercial services that cost quite a lot and considering they only work proficiently in big cities with a lot of user data and fixed infrastructure input, they may not be the best option for a some more rural users. The usefulness of these services is heavily reliant on their reaction times. They cannot always avoid congestion, since congestion can often be spontaneous due to an accident. Google Maps and Microsoft's Bing use statistical predictive analysis that suggest where congestion may begin and end, but due to the volatile and unpredictable nature of congestion and accidents, these tools are only useful to a certain extent. These predictive systems, which TomTom and Garmin also employ, rely on the recurring congestion trends, this only accounts for 50% of all congestion [17], so in order to determine the effect congestion has on traffic because of random incidents, simulation must be used. The system which we will build will simulate incidents and test the performance of re-routing vehicles based on the detection of these incidents.

An alternative to our work would be the research done by Juan Pan *et al* on *Proactive vehicle re-routing strategies for congestion avoidance* [18]. In their paper they outline how they would implement their system model. It is composed of a centralized traffic monitoring and re-routing service, distributed across several servers and a vehicle software stack for periodic traffic data reporting and for showing drivers alternative routes, this would be an embedded system or installed on a smart phone. The re-routing strategies they employ make use of travel time as the only factor for computation of the shortest path. The three re-routing strategies they discuss are Dynamic Shortest Path (DSP), which finds the path with the shortest travel time and assigns it to the vehicles selected to be re-routed. They also discuss the Random k Shortest Paths (RkSP) strategy, which computes k shortest paths for every vehicle that has been selected to be re-routed, then it assigns at random, one of the k shortest paths to each vehicle. The main reason for using such a strategy is to avoid simply moving the congestion elsewhere, by balancing the load on many nodes. The third re-routing strategy that is proposed is the Entropy Balanced k Shortest Paths (EBkSP). When moving from DSP to RkSP the computation time increases, this is a problem since the time to transmit the detour to the user needs to be transmitted before they pass the re-routing intersection, so if the re-routing algorithm took too long it would be useless. This is why they introduced EBkSP which improves on RkSP, it performs a more intelligent path selection by considering the impact of every selection on the future density of the selected nodes. In addition to this, EBkSP ranks the cars that are to be re-routed on an urgency function, that ranks the vehicles that get more adversely affected by the congestion first so they get re-routed first. In order to avoid the problem of shifting congestion elsewhere, the

EBkSP algorithm associates a popularity measure to nodes that will be used in the future.

In order to implement a simulation of this model and evaluate the performance of the routing algorithms they make use of SUMO and TraCI. They implemented their routing strategies defined above using TraCI, so that when SUMO is run, it waits for a command from TraCI, when a command is received it runs the simulation for the specified amount of time. In their simulation they used maps of Brooklyn and Newark. They modified the default routing algorithm to find the shortest paths using the travel time metric instead of the default distance metric defined in SUMO. They ran their simulations for 15 runs and gradually reduced the total number of users that adopted the re-routing system in order to establish the affect it has when less people use the system. Below is a figure which illustrates the flow through the network which they employed. In the results they presented, there is

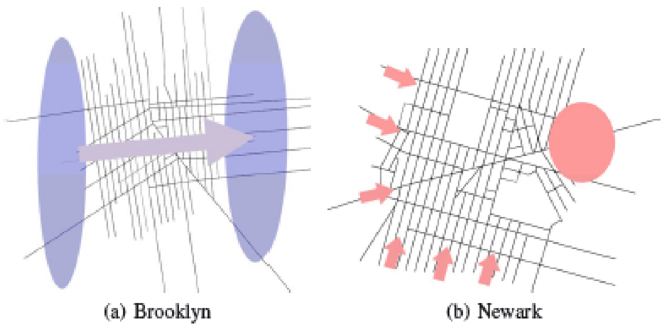

Figure 2.2: Flow of traffic on the network [18]

a significant decrease in travel time when implementing all three of the re-routing strategies compared to no re-routing and of these strategies there is a clear step down graph with EBkSP having the best performance. Compared to the simulations without re-routing there was an 81% decrease in travel time. They also assessed the level of compliance of the users, since it is unrealistic to assume all drivers in the network would have the system, but they found that even with a low level of compliance the re-routing strategies still worked quite well.

2.3 Missing Features

On the SUMO website they have a ticket system, whereby the users can add tickets which are jobs that need to be done, sometimes something as simple as bugs or typos, but amongst the bugs are suggestions for more substantial improvements. They also include a student and support page with possible suggested projects. Among the list of projects are topics related to traffic science, information science and other issues.

The suggestions related to Traffic Science include Pollutant Emission modelling and Evaluation, this would involve modelling the amount of emissions from given road networks based on a realistic traffic flow for that network throughout an average day or week. Another suggestion was a traffic light comparative study, this would involve testing various traffic light algorithms performance for certain traffic loads.

Regarding the route choice and demand modelling there is a lot more suggestions. The evaluation of one-shot traffic assignment, which involves assigning a route to each vehicle on the network at the start of simulation. Another route choice suggestion is to come up with alternative methods for shortest path search in large networks, where just using Dijkstra would be highly computationally intensive. They also have two suggestions for induction loops, which are to use induction loop values for Highway demand generation or to extrapolate routes based on these values.

On this page they also suggest forecasting of Demand Time-Lines which would involve running simulations on real networks and using realistic traffic flow on the network, to work out at what times does traffic flow on certain nodes on the network become too much to the point where it causes congestion.

Within their traffic simulation models section they suggest concrete validation of simulation, this would be a huge task to be undertaken by a team like the Tapas Cologne project [20]. Another simulation model suggestion would be to simulate emergency service vehicles. This would be a very useful simulation, since during rush hours, accidents often occur, and with road networks at their maximum capacity in most cases, it is very difficult for emergency services to reach their destination quickly. If you could run simulations for hundreds of destinations on a network from a hospital or police station it would surely prove as a great source of information for creating a specialized GPS system for emergency vehicles.

Another missing feature which would greatly affect congestion alleviation in SUMO is making use of the opposite lane for overtaking, since at the moment a vehicle is stuck on its side of the road, if a vehicle gets stopped behind a stalled or crashed car then it is essentially trapped, whereas in reality it would just overtake and keep going. In some cases in SUMO a car can over or undertake a crashed vehicle if the road they are on has more than one lane going in the same direction e.g. dual carriageway.

There was a problem that was encountered when initially importing maps into SUMO, Irish maps in particular, was the fact that SUMO is right hand traffic only, being of German design that is what you would except, but they hope to support left hand traffic in the future. One of the main reasons we felt there would be issues with simply running a simulation with right hand traffic is the use of filter lanes and other lane division systems that are part of the road network in Ireland may not work correctly.

Lastly, under their simulation models section, they mention pedestrian flow, they would like to see this included, since during rush hours, pedestrian traffic has an effect in and around train station and bus stops. Where footfall would be very high, traffic lights have to alternate more often thus slowing down traffic. This would lead to a more accurate simulation.

One of the major missing features that was found while getting to understand SUMO is how the route definitions are all made prior to runtime by default. In order to update these routes you have to implement TraCI alongside SUMO so that the routes can be redefined during runtime. Anther missing feature is the way the route is calculated. As per the paper about Proactive re-routing, SUMO should have more than just the shortest path to calculate the route. At the moment it only takes into account the shortest path from the source node to the destination node.

Chapter 3: **Investigating the Impact of Accidents on Road Traffic Congestion**

3.1 Simulation tool

The simulation tool we used to simulate a road network and demand on that network is SUMO [13]. This simulator allowed us to generate traffic demand on a small grid network (using NETGENERATE) to begin with, enabling us to see what was happening during runtime.

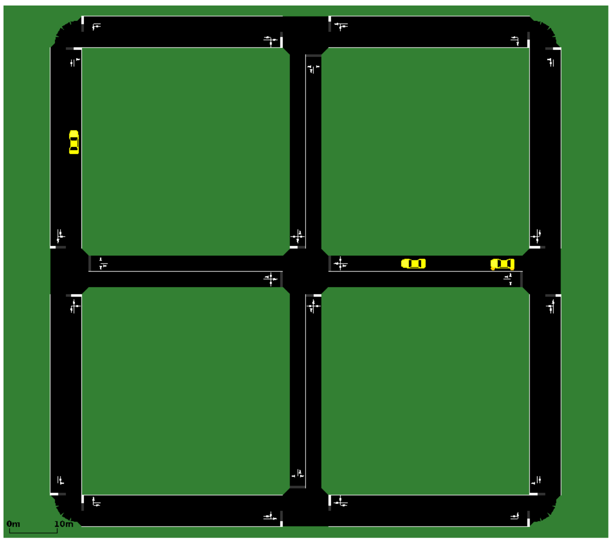

Figure 3.1: Simple 3x3 Grid Network

This was essential because OpenStreetMaps are often too complicated to see exactly what is happening. This small scale map was used to build our initial experiments on, by first creating an accident.

3.2 How to create an incident?

An incident refers to anything that stops the flow of traffic. Examples in the real world could be a collision, road works, bad weather causing very slow speeds, or just sheer volume of traffic. There are a few options in SUMO to simulate such an event. Stopping a car for a fixed period of time, by defining a point along its route when it should halt and for how long. Manipulating traffic lights so one stays red for a longer period of time than it should. Or setting the speed limit on an edge so low that the traffic is practically not moving. We decided to employ the first option, since stopping a car was the easiest of the three to implement and therefore easy to port from one scenario to another. In order to stop a car on a given edge we first need to establish the edge and lane IDs. This is quite straightforward using the SUMO-GUI by right clicking on any edge, which gives the lane ID, from here you look up the edge list and include the following code in the vehicle definition that you want to stop to simulate a traffic jam.

```
<stop lane="0/0to1/0_0" endPos="10" duration="200"/>
```

This code stops the vehicle on the specified lane at 10 meters for a duration of 200 seconds.

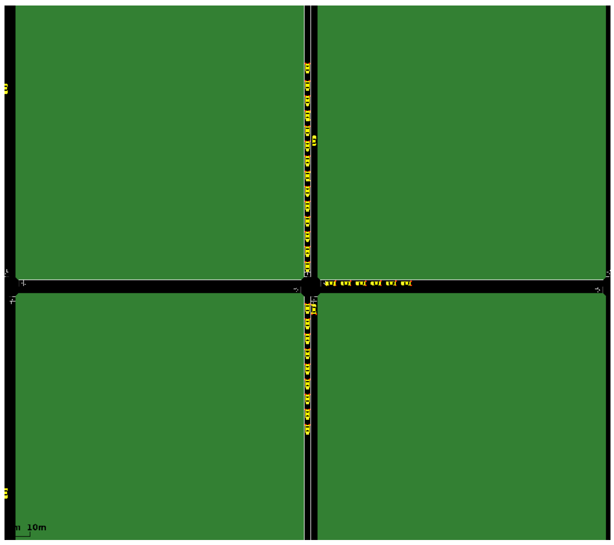

Figure 3.2: Car causing a jam

In the image shown the car at the bottom of the screen is being halted, which in turn stops all cars that are behind it or trying to get onto the route.

Various problems arose when it came to initially getting a car to stop. Figuring out how long to halt for a realistic traffic accident. From driving experience on the road (not in accidents) we figured ten to twenty minutes would be sufficient to impact on the traffic on the route, as well as being a realistic time in which the vehicles involved are left blocking a road, obviously more severe accidents can take hours to clear but in reality these more serious accidents tend not to happen in low speed urban environments.

Another issue we had when stopping a vehicle was teleporting. In SUMO when a vehicle waits for more than 300 seconds it teleports by default. This means that when a car is stopped to simulate an accident if we want a realistic time of 1200 seconds only the first 300 seconds will stop the vehicles behind and then they will teleport. In SUMO a car teleports by going "under" the road, and popping back up at the first chance it gets when the road ahead is clear. In order to overcome this the time to teleport was increased to 1200 seconds, which led to yet another issue.

As it turned out teleporting was a bug prevention measure to tackle random collisions, in SUMO collisions aren't technically defined, but are observed. To overcome them the two or more vehicles involved are teleported. Since the time to teleport was increased to establish more realistic travel times, any collision that occurred further increased traffic jams and caused congestion that could not be prevented. These collisions were mainly due to traffic light issues and road networks not being properly converted using SUMO's NETCONVERT tool (used to convert OpenStreetMaps to SUMO networks), which is out of my quite difficult to alter with limited time, so instead we decided to use more straight and grid-like networks of US cities, instead of Dublin.

Other arguments can be made for not using SUMO to simulate Irish or left-hand road networks in general. The main reason is that SUMO is by default a right-hand road simulator. When a left hand road network is fed into NETCONVERT it gets most of the connections right. However, it cannot handle roundabouts, instead of inverting the direction it remains as is and sends the traffic the left way around instead of right. It also cannot handle one way streets or dual-carriageways or motorways correctly, since these roads all have explicit directionality, it goes with whatever this definition is and sends the traffic down the road the wrong way. This might seem trivial but when you have more than one lane of traffic trying to essentially swap sides of the road it gets messy and collisions occur.

Evaluation results

When the simulation was run without any manually inserted incidents, a fixed travel time would be generated in SUMO. To output this data in the form of a file some minor configurations had to be made. Within the output file it lists various metrics of the route, departure time, arrival time, trip duration etc. this is in XML, which needed to be parsed. Initially the simulation was run with no accident and the trip duration data was written to a file, then an accident was introduced and that new trip duration was written to a separate file. In order to compare the two results a HashMap was required, using the VehicleID as the key. This way the no-accident data could remain the same whilst various additional tests could be run, such as modifying the number of accidents. Only the vehicles that returned an increase in travel time of more than 2 minutes were considered, this was done to reduce the error rate. Some random vehicles would have an reduced travel time and also other vehicles had a mimimal change of less than 2 minutes, some other vehicles had seemingly ridiculous times so any times above 4000 seconds were discarded. These were chosen as cut off points because on the lower end it is 10% of the accident time, and 4000 seconds sets the upper bound high enough to eliminate outliers, which proved to be a more realistic representation of what could be seen on the network when running the simulation. These bounds returned a subset of the most affected vehicles, and these were the ones taken into account for the results.

The graph below illustrates the effect that the number of accidents has on the travel time of vehicles. The times shown are the average increase in travel time with respect to the trip duration when no accident occurs, this value is calculated as shown in the formula below.

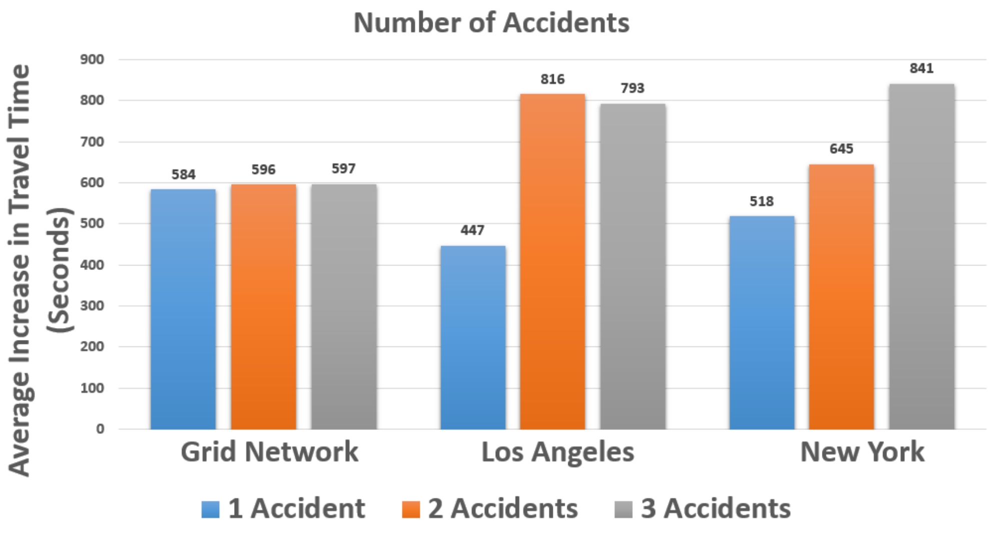

Figure 3.3: Impact of the number of occured accidents on the increase in travel time

$$AverageIncrease = \frac{\sum_{i=1}^{n} TT_{\text{increase}}(vi)}{n}$$

$$\text{where } TT_{\text{increase}}(vi) = TT_{\text{afterAccident}}(vi) - TT_{\text{beforeAccident}}(vi)$$

$$TT = TravelTime, vi = \text{every affected vehicle}, n = \text{number of affected vehicles}$$

In the three scenarios shown, the Grid Network is a 10x10 Grid Similar to the one shown above, Los Angeles is an extract of the area in and around Hollywood and New York is the area of Lower Manhattan. The Grid Network has such consistent results because there are no traffic lights just crossroads LA was more volatile because of the nature of the topology, it varies from very wide multi-lane roads to small single lane each way traffic. New York like the Grid was also consistent since New York is essentially a Grid, but since it has more restrictive movement with lots of traffic lights the travel times increased as the number of accidents increased.

The graph below illustrates the ratio between the average increase in travel time and the average trip time without an accident. The Grid Network had very short trip durations without an accident, this meant the 1200 second accident had a greater percentage increase than the other two scenarios. New York's' average travel time was almost half an hour, so a twenty minute delay did not have as much of an affect on the overall average increase.

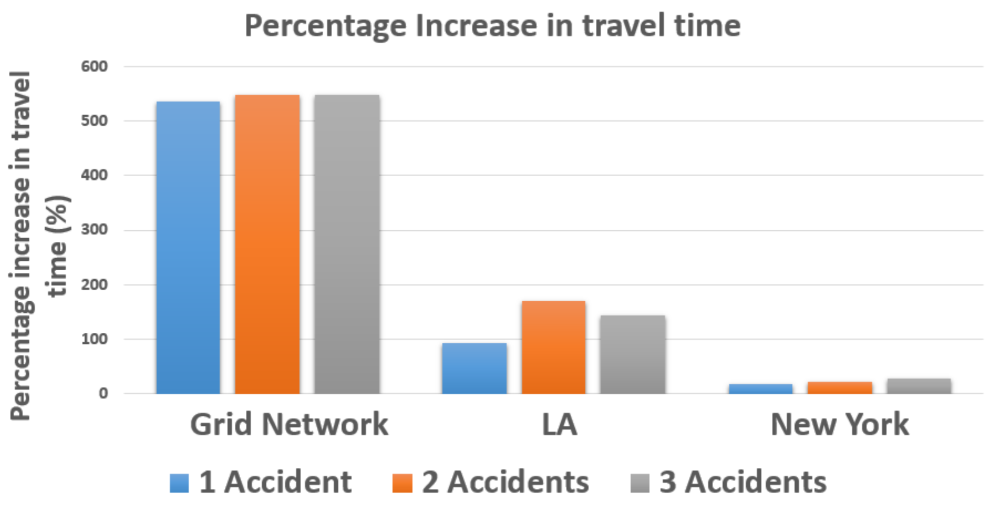

Figure 3.4: Ratio between increased travel time and average travel time without an accident

There was a lot of trial and error when it came to creating these accidents and trying to garner some sort of baseline from which the testing could be done. This trial and error mainly consisted of accident location as well as the length of time of an accident and the time in which the accident occurs. All of these factors directly affect the outcome. All of the above results have a fixed vehicle density between tests, of 3600 cars per hour. The length of time of all of the accidents was also fixed, at 20 minutes. The location and timing of accidents was where the variation comes into play, I will illustrate with the 10x10 grid network how these factors affect travel time.

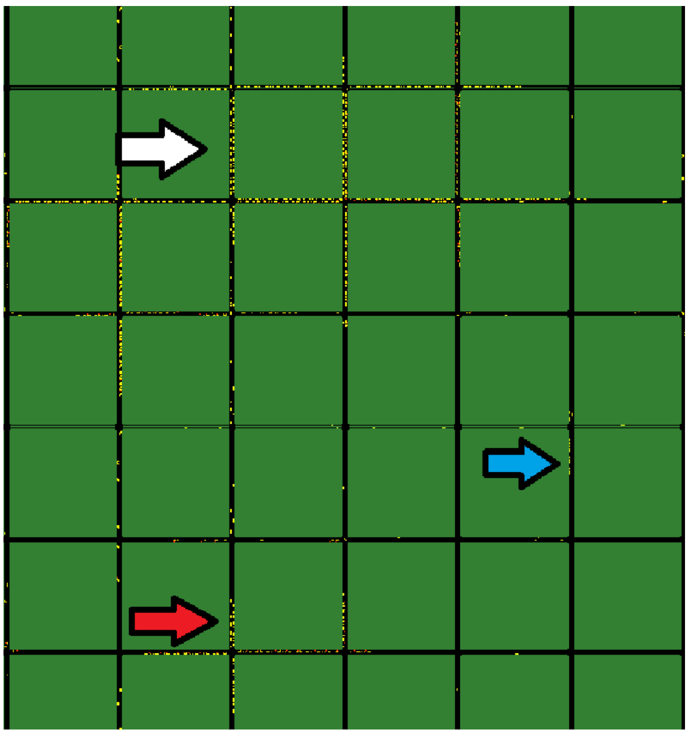

Figure 3.5: Order of Accidents: White - 1st, Red - 2nd, Blue - 3rd

The image above illustrates how the location and timing of the accidents affected the overall increases in travel time. The biggest jam is at the 1st accident location (every yellow dot is a car), therefore less traffic has the opportunity to get to the other accidents. This explains the lack of a major jump in travel time with respect to the number of accidents occurring in the Grids results.

Another experiment that was carried out was to compare the effect of traffic density on the overall increase in travel time. In the graph shown below for each scenario, the accident data from above was averaged and used as an overall average increase in travel time with respect to the traffic density.

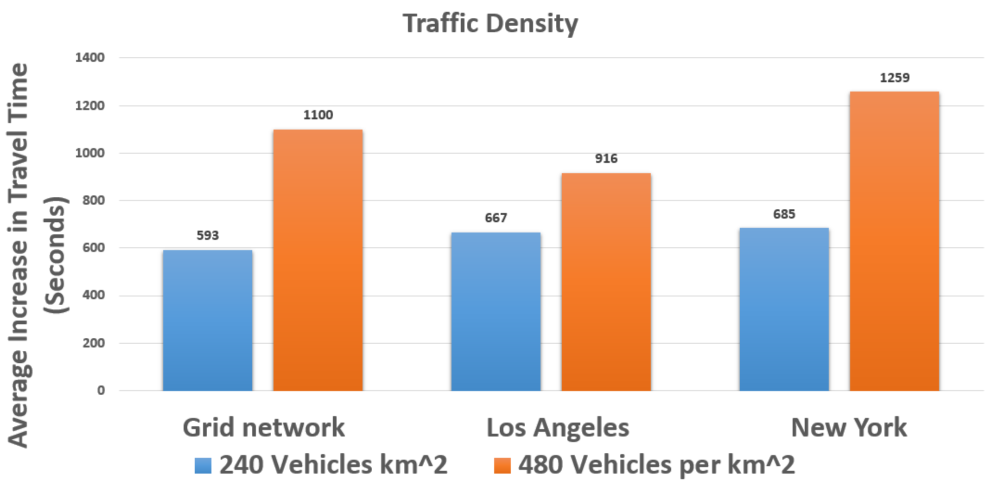

Figure 3.6: Impact of traffic density on Travel Time

The density was then doubled to 480 cars per km^2 and the same tests were run. From the graph you can see that the doubled traffic density has a similar affect across all scenarios.

The Grid and NY had the highest jumps because they have short streets with a lot of traffic lights and junctions. Whereas with LA there was less of an impact due to the greater number of lanes on each road and the less frequent traffic lights. From running the simulations it became clear that the higher the traffic density the fewer cars made it out of the simulation in the same time as the lower density (simulation time was run for 4500 seconds). In order to overcome this issue the simulator was allowed to run until all cars left the simulation.
As you can see from the results above the topology affects the increase in travel time also, below is a graph illustrating this effect by aggregating all simulation results from previous tests.

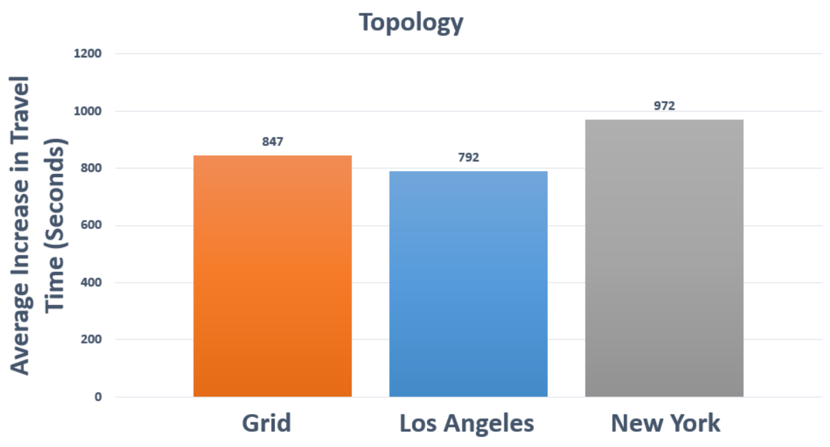

Figure 3.7: Impact of topology on the increase in travel time

As one would expect the grid and New York were the two which were most adversely affected by their topology, since they both have a lot of single lane roads and in NY there are mostly one way streets. In comparison to LA which as mentioned has very wide streets and very few one way streets.

The final comparison that had to be done was the affect that the duration of an accident has on the overall increase in travel time. In order to carry out this test the duration of the accident was tripled, to 60 minutes.

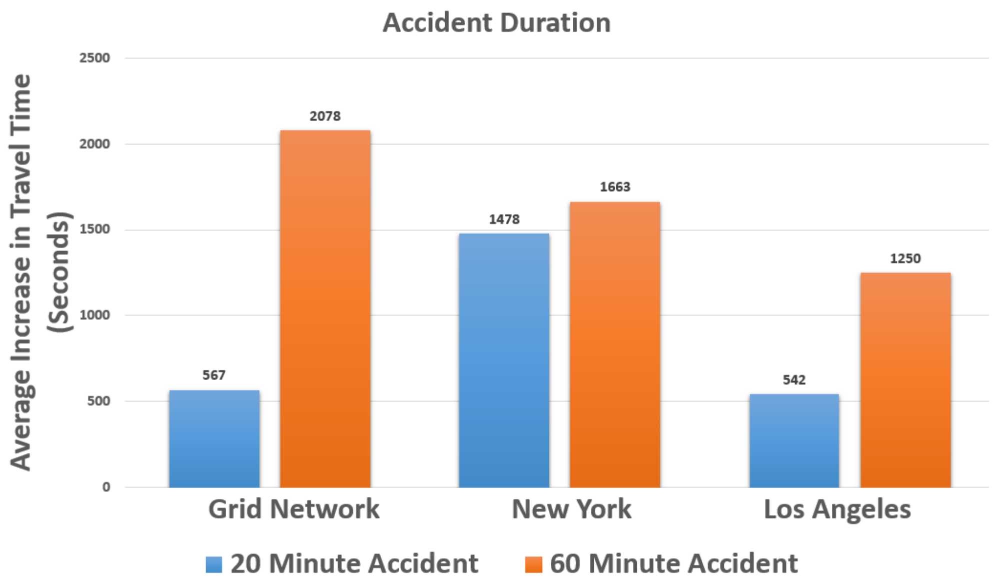

Figure 3.8: Impact of accident duration on travel time

This data was established by averaging the increase in travel time for three accidents for both accident durations. From the graph it is clear that the accident duration further increases travel time, but the topology of the networks really comes into account here. The Grid Network simple became jammed, almost entirely when an accident lasted for so long. New York since it is similar to the Grid had the same problem, but since there was such gridlock on the network to begin with, the increase in accident duration had less of an effect. LA as usual had a more reasonable increase since its streets were not as gridlocked as the other two. The significance of these results will allow us to test our solution which is explained in the next chapter.

Chapter 4: **Re-routing Mechanism Design**

4.1 Key Principles

There were two options to reroute vehicles in SUMO. The first was to use statically defined routing that is provided in SUMO. This static method is deployed by adding a re-routing file to SUMOs configuration. All vehicles that are to be re-routed have to be listed in this file prior to runtime. This method was tested using very small scale scenarios, in which it worked well. However, since all the vehicles had to be defined prior to runtime it was not feasible to use this approach, the main reason being that in other scenarios there would be thousands of cars. This static method could be made dynamic by writing a script perhaps but the second re-routing option seemed more appropriate for this project.

Combining SUMO with TraCI was the other option available, it involved using the server-client connection between the two and using the TraCI API in python to alter the state of vehicles during simulation runtime. No vehicles had to be defined beforehand, only the road which was going to be used as the accident and the trigger roads surrounding the accident road.

When initial testing on this re-routing with TraCI began, simple small scale tests were done, where the vehicles routes were manually defined so the actions of the specific vehicles could be monitored. Once the strategy was working correctly a more dynamic approach was needed in order to perform large scale and multi scenario tests. The solution is explained below using the transition diagram.

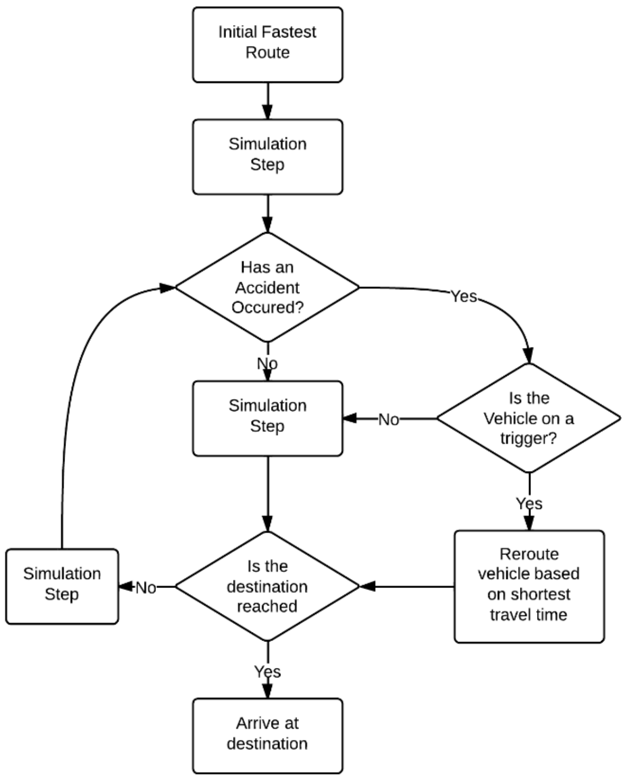

Figure 4.1: Flow Diagram illustrating the proposed re-routing strategy

The trigger mechanism was used to make the solution less complex, it works by adding the roads that are connected to the accident lane to a list, and then re-routing the list of vehicles on the trigger lanes at each simulation step. The complexity of our mechanism was of order N, where N is the number of vehicles on the trigger lanes in each simulation step. The alternative to this approach was to only reroute the vehicles that had the accident lane in their route definition. In order to deploy this alternative approach, for each vehicle on the network all their routes had to be written to a list (some routes containing up to a hundred edges), and then checking if each vehicle contained the accident lane in its route. The complexity of this approach was quickly realised when the simulation steps exponentially increased in time as more vehicles deployed onto the network ($O(N^2)$). Below is a snapshot illustrating how the trigger mechanism works.

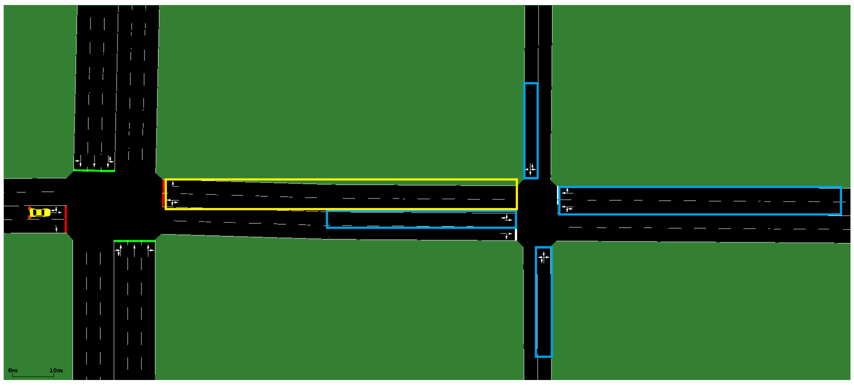

Figure 4.2: Trigger Lane Mechanism

In the image above the blue boxes are showing where the trigger lanes are, the yellow box is the accident lane. When a car is found to be on a trigger lane its route is recalculated. During its recalculation the accident lane is excluded because it has a negligible speed limit (the speed limit is set to 0.001 meters per second to simulate a lane closure). The beauty of this approach is that it will not alter the route of vehicles that are on a trigger lane but don't intend to go onto the accident lane, because when the vehicles route is being recalculated, it doesn't have the accident lane in its route to begin with, so the re-routing will just return its original route.

Below is a python implementation illustrating how this mechanism works.

```
def run(laneId,accidentSpeed,normalSpeed,step,Start,Stop,trigger):
    if step <= stop and step > start:
        createAccident(laneId, AccidentSpeed)
        monitor(trigger,laneId)
    else if step > stop:
        stopAccident(laneId, NormalSpeed)

def createAccident(laneId,accidentSpeed):
    setMaxSpeed(laneId,accidentSpeed)

def stopAccident(laneId,normalSpeed):
    setMaxSpeed(laneId,normalSpeed)

def reroute(vehicle):
    rerouteTraveltime(vehicle)
    setColor(vehicle, (255,0,0,0))
```

```
def monitor(trigger,laneId):
    for edge in trigger:
        vehicleList = getListofVehiclesonEdge(edge)
        for vehicle in vehicleList:
            reroute(vehicle)
```

The run function is called after every simulation step, within this function if an accident is in progress then the monitor function is called. The monitor function checks all trigger lanes to see if there are any vehicles on them, if there is the reroute function is called.

4.2 Performance Evaluation Results

To evaluate the performance of our mechanism, multiple tests and scenarios had to be run, similar to the accident testing in the previous chapter. For each scenario there were seven tests on a given traffic density. The seven tests involved were; no Accident (Just run the simulation), Accident no re-routing (For one, two and three different accidents) and Accident with Re-Routing (also for one, two and three accidents). For the initial full scale test, the 10x10 Grid Network was used.

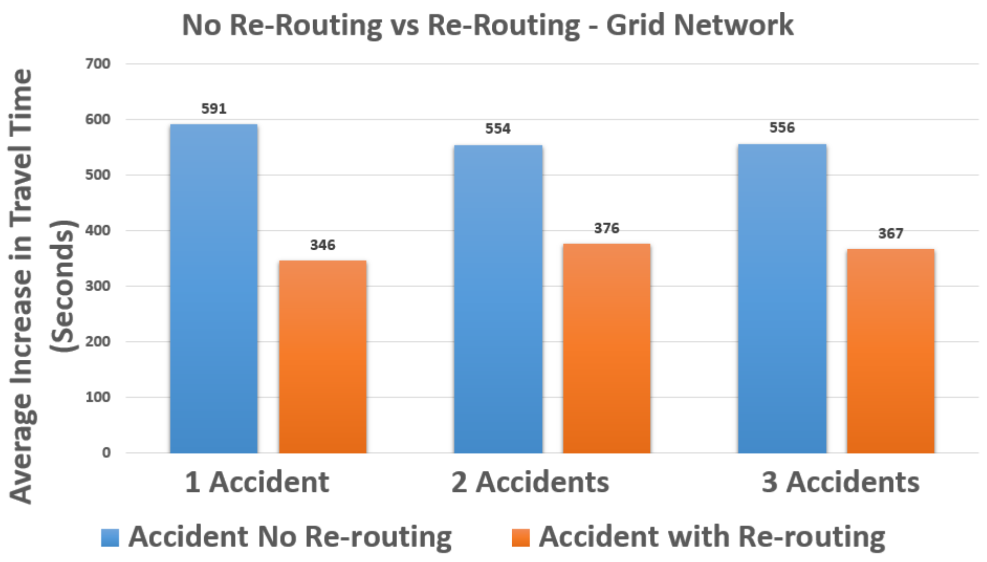

Figure 4.3: Grid Network Re-Router Performance

From the graph you can see how the best re-routing improvement came with one accident, mainly due to the small scale of the map. But the re-router still performed very well with two and three accidents. The overall average improvement of the re-router on the Grid Network was 35%. Meaning the average ten minute delay would be brought down to a 6.5 minute delay.

The second test that was carried out was LA, where the most positive results were collected. With its wide lanes and lack of one way streets it was the perfect scenario for the re-router.

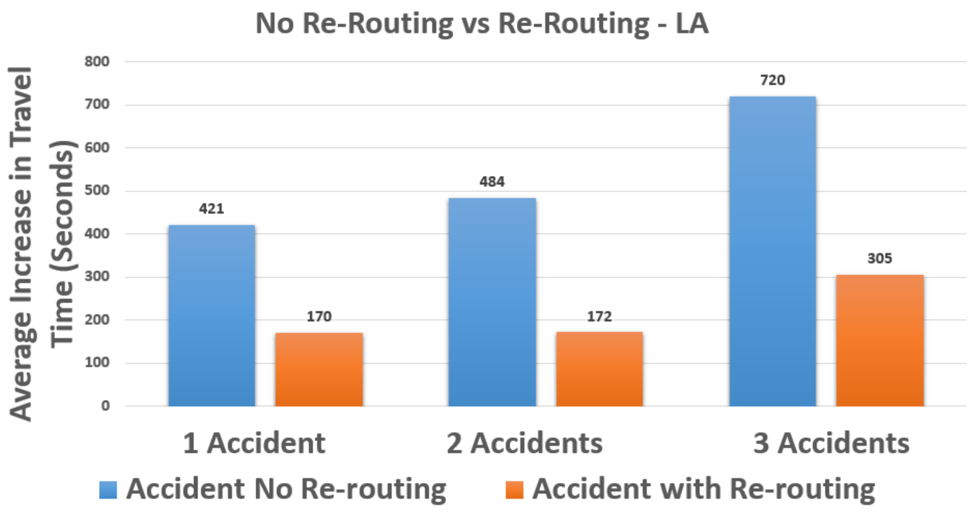

Figure 4.4: LA Re-Router Performance

The results clearly indicate that the re-router was significantly quicker than the accident simulation, overall on average it gave a 60.5% improvement on the no re-routing scenario.

The third test was on New York, from the offset it was assumed that New York would not provide the same level of improvement as the Grid or LA because of the frequency of one-way streets. This meant that more often than not vehicles would be sent in the wrong direction just to get through the maze of one way streets to get to their destination or back on their original path. The results proved this assumption.

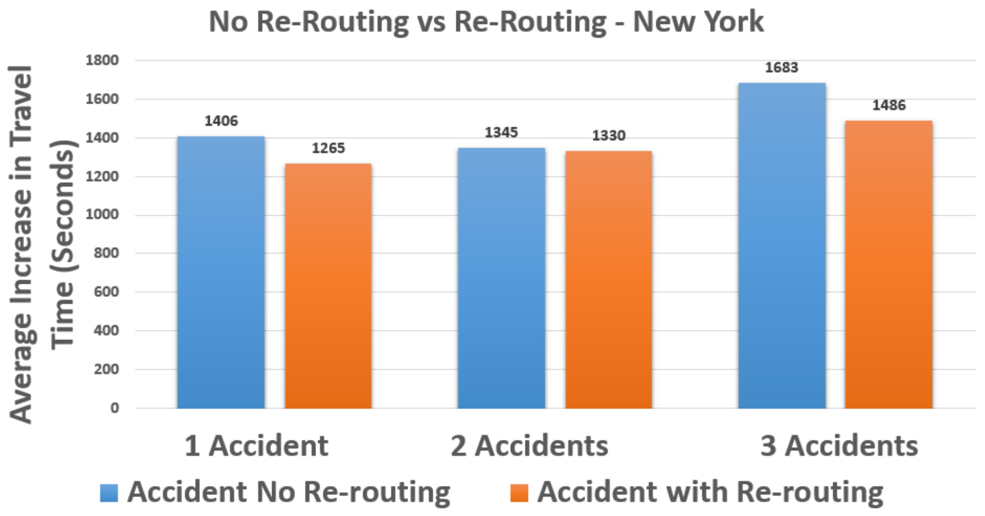

Figure 4.5: NewYork Re-Router Performance

Whilst the results were not as good as the previous two, they still showed an overall improvement of 7.6%. This could be improved on by altering the way in which vehicles were re-routed, by employing a different algorithm which would into account the weight of certain nodes which are often beyond capacity in cities such as New York. If this weighting metric was added traffic could be diverted via the less frequently used nodes and alleviating congestion overall. It was found that in the NY simulation in particular, there were many collisions. As mentioned previously, collisions are bugs that are handled by teleporting, but there seemed to be an inordinate amount in NY, which caused further increases in travel time. This definitely affected the results of the re-router because vehicles would often get stuck behind a car that randomly collides with another, with no way of getting free from the jam it must wait until it is teleported. This increases the overall average increase in travel time because if a vehicle was re-routed and then it gets stuck in a jam (nothing to do with the manually inserted accidents), its travel time closely approaches or exceeds that of one which simply waits for the accident it was diverted from, to be cleared.

Our solution worked well on the above scenarios, but all the above tests used a fixed relatively short accident duration of 20 minutes. To further test the efficiency of our solution the accident duration was increased to 60 minutes. This meant that the re-router would have the chance to re-route much more traffic.

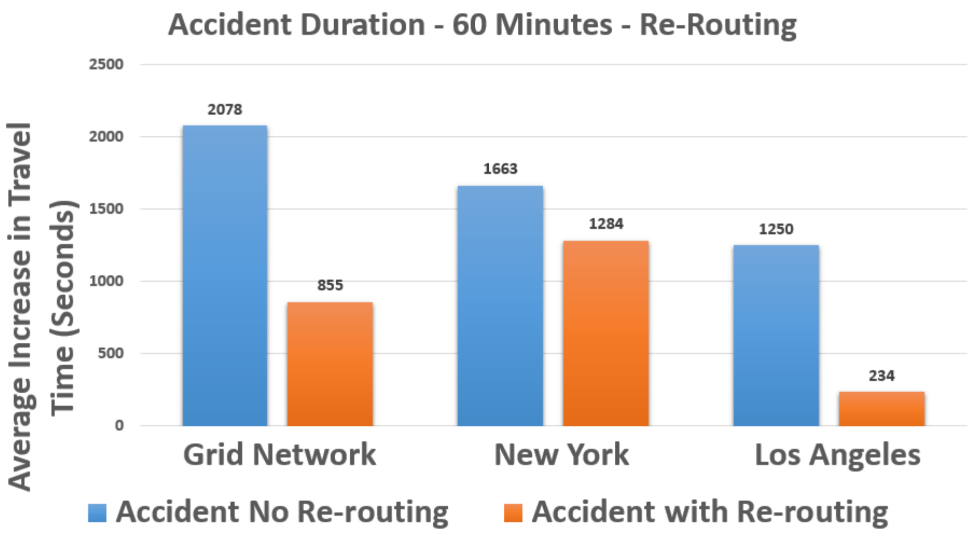

Figure 4.6: Re-Router Performance 60 Minute Accident

The results shown in the graph were obtained by running the same number of accidents per scenario as before, and averaging the increase in travel time from all accidents. The performance of the re-router on the 60 minute accident was better than the 20 minute accidents in all three scenarios, providing an overall average reduction of 53%. This was a 50% improvement on the average reduction in travel time from the 20 minute tests, which was 35%. These tests proved the theory that the longer an accident is the more beneficial it is to try a different route. This parallels a real world scenario to a degree. If a driver was informed of a traffic jam that would last half an hour, the driver would be much better off to go another route that may add 20 minutes to the journey. The only thing that the driver does not know is whether or not an accident has occurred on the detour road they chose to take. This is what leads to further increases in travel time, which often happened in our simulations. To overcome these errors we ran the accident for a longer duration, to illustrate the effectiveness of our solution. The main flaw with re-routing traffic is that the detour must take less time than the accident. If the accident is too short (less than ten minutes), more often than not the driver will end up worse off than if they decided to wait in the traffic jam.

From all the results above, it is clear that topology has a major role in the effectiveness of our re-routing solution. The more one way streets, single lane 2-way streets, traffic lights and junctions means the more complex a detour becomes. The graph shows the average reduction in travel time based on the given topologies.

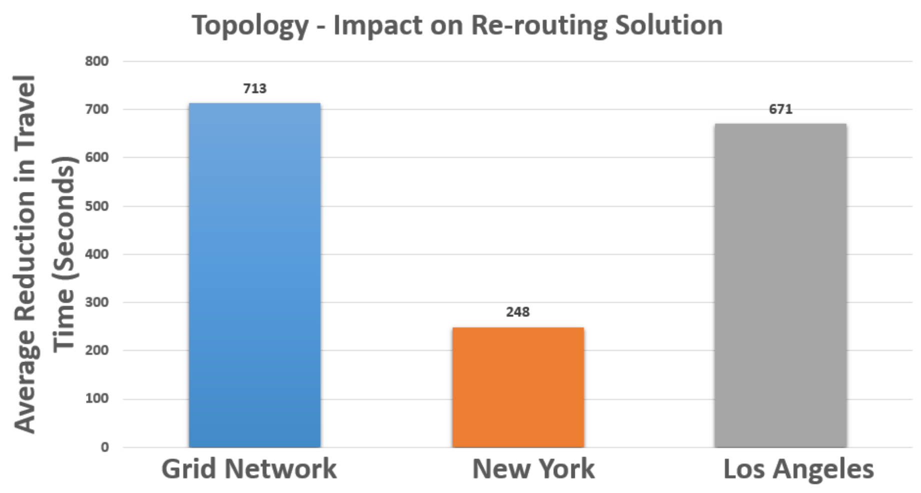

Figure 4.7: Illustrating the effect network topology has on our solution

Even with a small sample of different topologies, major differences can be seen in the effectiveness of re-routing. This reflects the real world equivalent cities. As you would imagine it is easier to drive around LA than NY, and not just because of the number of cars on the road, but because of the shape and nature of the road networks.

Chapter 5: **Conclusion**

The main goal of this project was to extend SUMO by implementing a mechanism which dynamically updates the vehicles' route during simulation runtime. This has been achieved through TraCI and its python API which was coupled with SUMO.

After running numerous initial tests to highlight the effect of accidents on travel time, our mechanism which updated the vehicles' routes during runtime was developed and further tests were done to evaluate the performance of our proposed design. These results illustrated that the re-router reduced the overall increase in travel time by an average of 35% when the accident duration was fixed at 20 minutes. However, when the accident duration was increased to 1 hour, the re-router performed 50% better, with an average reduction on the increased travel time of 53%. The topology had a major impact on the effectiveness of our mechanism, this reflects all real world scenarios where the topologies of road networks affect traffic. The re-router performed best on Los Angeles, with a mean reduction on the increase in travel time of 75%, it performed worst in New York when the accident duration was set to 20 minutes, with a mean reduction of 7.6%. This difference comes down to topology, vehicle density and the overall quality of the simulator. Bugs plague SUMO, mainly because it is open source and not commercial, so getting reliable consistent results wasn't always easy.

An extension of our work could be done to make the re-routing more dynamic by allowing accidents to be created on the fly and only decide on their duration. Other routing heuristics could also be considered if more time was available during the project, route selection metrics such as travel costs or ease of driving could also have been implemented. A travel cost metric would take into consideration the fuel consumption of the route and the toll costs (if any) of the trip. The Ease of Driving metric would be a more complex one, consisting of things like the number of lanes, frequency of traffic lights and junctions, the width of the roads and even the speed limits. These heuristics would have to be manually implemented using SUMO and TraCI which was not possible with the given time constraints of the project. A further extension of SUMO could be done which would use VANETs (Vehicular Ad-Hoc Networks) to control the vehicles mobility during a simulation. This would involve deploying an virtual network (using ns-3 [21]) of vehicles that could communicate with each other about the conditions present on the roads. For example if an accident were to occur the car closest to the scene would send a message to all other cars that are near it, telling them to avoid the road and to pass on the message.

To conclude, our solution has worked well on some of the scenarios it was tested on, but not all, in the future it is our hope that this solution could be improved so that any scenario on any map produces a similar level of performance.

Bibliography

[1] R. Arnott & K. Small (1994), *The Economics of Traffic Congestion*, American Scientist, Volume 82(5), 446-455.

[2] `http://www.transport2020.org/`

[3] P. Goodwin (2004), *The Economic Costs of Road Traffic Congestion*, Rail Freight Group, London UK, May 2004.

[4] S. Joerer, C. Sommer, and F. Dressler. (2012) *Toward Reproducibility and Comparability of IVC Simulation Studies: A Literature Survey*, VANET '12. ACM, NEW YORK, USA, 27-32.

[5] `http://www.tomtom.com`

[6] `http://www.garmin.com`

[7] `http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/`

[8] `http://www.paramics-online.com`

[9] `http://www.aimsun.com/wp/`

[10] J. Harri, M. Fiore and F. Fiali and C. Bonnet, *Vehicular Mobility Simulation with VanetMobiSim*, Trans. Soc. Modeling and Simulation, 2009.

[11] D. Choffnes & F. Bustamante *STRAW - An Integrated Mobility and Traffic Model*, In Proc. of the 2nd ACM International Workshop on Vehicular Ad-hoc Networks (2005).

[12] `http://wiki.openstreetmap.org/wiki/TIGER`

[13] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. *Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements*, 5 (3&4):128-138, December 2012.

[14] A. Wegner, M. Piorkowski, M. Raya, H. Hellbruck, S. Fischer and J.P. Hubaux. *TraCI: An Interface for Coupling Road Traffic and Network Simulators*,Proc. 11th Comm. and Networking Simulation Symp. (CNS 08), Apr. 2008.

[15] V. Ngoc Nha, S. Djahel and J. Murphy, *A Comparative Study if Vehicles' Routing Algorithms for Route Planning in Smart Cites*, VTM 2012, Dublin, Ireland, 20 Nov 2012.

[16] `http://www.inrix.com/`

[17] B. Coifman and R. Mallika. *Distributed surveillance on freeways emphasizing incident detection and verification*, Transportation Research. Part A, Policy and Practice. Volume 41(8), 750-762.

[18] J. Pan, M. Khan, I. Popa, K. Zeitouni and C. Borcea. *Proactive vehicle re-routing strategies for congestion avoidance.*, 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems, Zhejiang, China, 16-18 May 2012.

[19] `http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Students_And_Support`

[20] http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Data/Scenarios/TAPASCologne

[21] https://www.nsnam.org/