iTranSIM

-

Simulation-based Vehicle Location

Tino Morenz

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

2007

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Tino Morenz

2007-09-14

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Tino Morenz

2007-09-14

Acknowledgements

Many thanks to my supervisor, Dr. René Meier, for his guidance and support throughout and beyond this project.

Thanks also to Daniel Krajzewicz for patiently answering all my questions about SUMO. Furthermore thanks to the people in the SUMO community, for all the effort they put into that project.

Special thanks to Arthur Doohan for the long discussions and his encouragement. Thanks to the entire UbiCom class for a great year.

Finally, thanks to my parents for making all this possible.

Tino Morenz

University of Dublin, Trinity College

September 2007

Summary

Public transport provides affordable, 'eco'-friendly transport in urban environments. With the advancements in computer technologies real-time passenger information (RTPI) are now available for passengers, helping them to plan their trips so they spend as little time as needed travelling.

To provide real time passenger information it is necessary to have information on vehicle positions, so that travel times can be calculated and presented to the passenger. The iTranSIM system aims to provide vehicle locations not based on physical measurements but solely based on statistical data. Instead of locating the real vehicle e.g. using GPS, a virtual vehicle is tracked in an online simulation of Dublin City. The online simulation can be supplied with real traffic data such as vehicle counts from induction loops to create a realistic model of the traffic.

To demonstrate the feasibility of this approach the iTranSIM system was designed and a proof-of-concept was implemented based on the iTransIT framework. This framework was developed by the Distributed Systems Group (DSG) at Trinity College Dublin and provides means of integrating different components of an intelligent transportation system (ITS) so that data can be exchanged between the systems. To simulate the traffic the microscopic traffic simulator SUMO (Simulation of Urban Mobility) is used. SUMO is an open source project, maintained by researchers at the Centre for Applied Informatics at the University of Cologne and the Institute of Transport Research at the German Aerospace Centre.

The accuracy of vehicle location based on simulation could not be evaluated yet because a vital part of the necessary infrastructure is not yet available: the vehicle count data from real induction loops. Thus all experiments described in this thesis are based on empirical traffic data, however as soon as live data becomes available the accuracy of the proposed method can be analysed without any changes to the proposed system. Possible means of performing such an evaluation are also included in this dissertation.

Table of Content

Tables and Illustrative Material 1
Listings 2
Abbreviations 3
1 Introduction 4
1.1 Goal 4
1.2 Motivation 5
1.3 Approach 6
1.4 Thesis Contribution 7
1.5 Thesis Structure 7
2 State of the Art 8
2.1 Intelligent Transportation Systems 8
2.1.1 Advanced Public Transport Systems 9
2.1.2 Automatic Vehicle Location 10
2.1.3 AVL via Traffic Simulation 13
2.1.4 APTS and AVL in Dublin 14
2.1.5 iTransIT 16
2.1.5.1 Architecture 16
2.1.5.2 Spatial Programming Model 18
2.1.6 Conclusion 20
2.2 Simulation 22
2.2.1 Models 22
2.2.2 INTEGRATION 24
2.2.2.1 Model 24
2.2.2.2 Implementation 25
2.2.2.3 Summary 26
2.2.3 Cellular Automata 27
2.2.3.1 Model 27
2.2.3.2 Implementation 28
2.2.3.3 Summary 30
2.2.4 SUMO (Gipps / Kraus) 30
2.2.4.1 Model 31
2.2.4.2 Implementation 31
2.2.4.3 Summary 34

2.2.5 Conclusion 35
3 Design 36
3.1 System Design 36
3.2 Data Model 37
3.2.1 Road Network 37
3.2.1.1 Components 37
3.2.1.2 Data Model 41
3.2.2 Traffic 41
3.2.2.1 Components 42
3.2.2.2 Data Model 44
3.2.3 Public Transport 44
3.2.3.1 Components 45
3.2.3.2 Stop Times 45
3.2.3.3 Data Model 49
3.3 Online Simulation 49
4 Implementation 52
4.1 iTranSIM-DP 53
4.1.1 Data Model Population 53
4.1.2 Static Data Update 53
4.1.2.1 Road Network 54
4.1.2.2 Static Traffic Information 55
4.1.3 Dynamic Data Update 57
4.1.4 Simulation Control 58
4.2 iTranSIM-DUS 59
4.2.1 Output Processing 59
4.2.2 Data Provision 60
4.3 SUMO 61
4.3.1 Calibrators in SUMO 61
4.3.2 Vehicle Type Probe 63
4.3.3 Remote Server 64
4.4 Proof-of-Concept Implementation 66
4.4.1 Information Retrieval Using the Spatial API 66
4.4.2 Route Generation 68
4.4.3 Simulation Control 69
4.4.4 Output Processing 70

5 Evaluation 71
5.1 Evaluation Setup 71
5.1.1 Road Network 71
5.1.2 Traffic Patterns 72
5.1.3 Bus Stop Times 72
5.1.4 Test System 73
5.2 Accuracy 73
5.2.1 Offline Simulation 73
5.2.2 Online Simulation 75
5.2.3 Vehicle Locations 76
5.3 Extensibility 76
5.4 Performance / Scalability 77
5.4.1 Spatial API 77
5.4.2 Simulation Control 78
5.4.3 Output Processing 79
5.4.4 Simulation 79
5.4.5 Conclusion 80
5.5 Evaluation Conclusions 81
6 Conclusion 82
6.1 Future Work 83
6.2 Different Simulation Models 83
6.2.1 Alternative Applications for Online Simulations 83
References 86

Tables and Illustrative Material

Figure 1, Q-Time - Electronic Display [12] 14
Figure 2, The iTransIT architecture [20] 17
Figure 3, Multiple Lanes in Cellular Automata Network [26] 28
Figure 4, iTranSIM system design 36
Figure 5, Legal turns on a junction in theory (left) and reality (right). 40
Figure 6, Road Network Data Model 41
Figure 7, Vehicle Data Model 44
Figure 8, Bus Data Model 49
Figure 9, Traffic Flow - Fundamental Diagram [30] 50
Figure 10, iTranSIM System Implementation 52
Figure 11, iTranSIM - Data Container 53
Figure 12, Remote Server Message [36] 64
Figure 13, Remote Server - Simulate Step [36] 65
Figure 14, Remote Server - Update Calibrator 65
Figure 15, OSI Grid System Ireland [6] 67
Figure 16, Map of the bus route 15 71
Figure 17, traffic levels over a full day 72
Figure 18, Simulated Bus Travel Times 74
Figure 19, Simulation Performance Times 80

Listings

Listing 1, Calibrator Configuration 61
Listing 2, Calibrator Definition 62
Listing 3, Vehicle Type Probe Configuration 63
Listing 4, Vehicle Type Probe Output 63
Listing 5, SUMO Route File 68
Listing 6, Shell Script for Route Extraction 68
Listing 7, Calibrator Route File 69
Listing 8, SUMO Control Timing 69
Listing 9, SQL Statement for Loop Count Retrieval 78

Abbreviations

API	Application Programming Interface
APTS	Advanced Public Transport System
ATIS	Advanced Traveller Information System
ATMS	Advanced Traffic Management System
AVL	Automatic Vehicle Location
CPU	Central Processing Unit
DSG	Distributed Systems Group
GNU GPL	GNU General Public License
GPS	Global Positioning System
GUI	Graphical User Interface
HGV	Heavy Goods Vehicle
ITS	Intelligent Transportation System
JDBC	Java Database Connectivity
PST	Passenger Service Time
PT	Public Transport
QBC	Quality Bus Corridor
RTPI	Real-time Passenger Information
SAPI	Spatial API
SAX	Simple API for XML
SQL	Structured Query Language
STIS	Smart Traveller Information Service
SUMO	Simulation of Urban MObility
TCP	Transmission Control Protocol

1 Introduction

Everyday about 400,000 people use the service of Dublin Bus, the main provider of public transport in the Irish capital, to travel through the city [1]. This can be seen as an example for all major and minor cities around the globe. Public transport is vital because it offers affordable, 'eco'-friendly means of transport in urban environments.

So as to provide a better service to their customers public transport providers offer special information services allowing the passengers to plan their journeys. Possibly the oldest of such services is the timetable that provides information on which routes are served by a public transport provider and the estimated times when vehicles are leaving the stops. Nowadays the timetables are not only available in printed form but also electronically e.g. over the internet. Furthermore the data provided to the passengers can be enhanced using data reflecting the current situation in the public transport system, often referred to as real-time or live data. By taking actual vehicle positions into account travellers can be informed about real waiting or travel times.

But not only providers of public transport try improve their services by offering more up-to-date information to their passengers. In order to manage the traffic flows dynamically and in an intelligent way cities operate Intelligent Transportation Systems (ITS). The ITS gathers relevant data from different systems deployed in the city, such as traffic counting or weather stations. This information is then used to guide the traffic, e.g. using adaptive traffic light switching or displaying the number of available parking spaces on electronic signs.

To test new scenarios such as different traffic light sequences, traffic simulators can be used. These programs provide simulations of the traffic flow within a road network (e.g. a city) thus giving traffic planers the ability to test different setups (e.g. unrestricted v. bus only lanes, different traffic light setups). Therefore it enables engineers to find the optimal settings for an existing system without the need for real world testing which is nearly impossible within urban environments.

1.1 Goal

The initial goal of this project was to investigate the feasibility of a travel time prediction service for public transport, in particular for Dublin Bus. Research showed that there are two main components to such a service: the localization of vehicles and the calculation of travel times based on that location data. Both elements have previously been covered by researchers and their studies have yielded usable results as such systems are already deployed all over the

world. Examples of vehicle location systems can be found in section 2.1.2, for a detailed overview over travel time prediction the reader is referred to Karbassi and Barth [2] and references therein.

Further research into the subject revealed that the main expense in setting up a travel time prediction service is the deployment of a fleet wide vehicle location system. Thus we decided to shift our focus away from developing a complete service. Instead the main goal of the project is to investigate cheaper, more efficient means of collecting vehicle locations.

Herein we will focus on leveraging data provided by an existing ITS infrastructure to examine the possibility to locate vehicles not physically but solely on the basis of historical and live data. The way to calculate the locations is to create an online simulation of the traffic and to track vehicles in that simulation rather than in the real world. The online simulation is supplied with the relevant traffic data by the ITS and approximates the real world the traffic situation in real time in the simulation.

The long term goal is to examine the feasibility of a statistical vehicle location based on traffic data and online simulations. The system could than supply this data to user services which in turn calculate travel times and provide these times to the passengers in public transport. Compared to existing vehicle location systems this system would provide the data necessary for travel time estimation at lower cost regarding both the deployment and the maintenance of the system.

1.2 Motivation

The motivation of this project is to increase the number of travellers using the public transport in Dublin so that the volume of traffic is lowered and people can commence their journeys in an efficient and yet eco friendly way.

Dublin Bus serves about 93% of the population in the Irish capital and provides about 70% of the public transport therein. [1] However with the growing number of privately owned vehicles the traffic flow increases leading to congestion and unpredictable travel times, especially during peak hours in the morning and afternoon. Where the heavy traffic has hardly any effect on the rail bound public transport there is a big impact on the reliability and punctuality of bus services:

> *An 84X [NB: one of Dublin Bus' express routes] took 20 minutes to travel the short distance from Eden Quay to Trinity College, a journey that could be made in half that time on foot.[3]*

That excerpt from an article in the Irish Times from June 2007 illustrates that the problem of congestion and long travel times is immanent in Dublin. This view is also backed by Dublin Bus itself; the following quote stems their annual report 2006:

> *Congestion remains a major concern to the company and its customers. During 2006 the average speed of our buses was lower than ever. A report by consultants BDO Simpson Xavier in 2005 showed that congestion was costing the company €60 million per annum. [1], p.4*

Printed timetables are ineffective when travel times vary heavily due to the traffic conditions. Worse though, Dublin Bus does not even provide proper timetables for most of their routes. Only the times when a bus is supposed to leave its first stop is given, so the passenger waiting at any intermediate bus stop has to calculate the time it will take the bus to arrive at this particular stop by himself. This information policy leads to a rather poor public opinion on Dublin Bus' punctuality and service in general.

So people stop using the public transport because they perceive the service as being unreliable. This decreases Dublin Bus' revenue, but it also causes another problem with far reaching consequences. Those people who stop using Dublin Bus' service usually switch to their private vehicles, because alternative public transport providers such as the DART light rail system or the LUAS tram system only cover a small part of the town. By using their own car people contribute to the ever increasing amount of vehicles in the city, causing even more traffic and therefore decreasing the timeliness of Dublin Bus' vehicles even further.

To stop this cycle a better information policy needs to be established for the passengers travelling with Dublin Bus. When the public perception of Dublin Bus' service is changed to the better this will increase the number of passengers using the buses and therefore not only increase Dublin Bus' revenue but in the long run also decrease the amount of traffic in the city.

1.3 Approach

The following approach will be used for this project:

The current state of the art in intelligent transportation systems will be studied, with an emphasis on advanced public transport systems, automatic vehicle location and the iTransIT system which is in use in Dublin. Furthermore different models for traffic simulation will be discussed to find an appropriate simulator for our design.

Based on the results of the state of the art survey a system will be designed to predict the location of vehicles based on an online simulation. A proof of concept implementation of our design will provide the basis to discuss the feasibility of our approach.

1.4 Thesis Contribution

The thesis makes a contribution towards the integration of ITS systems and traffic simulators. First we investigate the possibility to implement an online traffic simulation of Dublin using the iTransIT ITS framework. Based on this online simulation the feasibility of a simulation based vehicle location system is explored and possible evaluation scenarios for such a system are described. Using such a combination to predict vehicle positions in a real-time manner is, to the knowledge of the author, unique and has not been proposed in the ITS community by now.

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2: State of the Art - This chapter gives an overview over existing vehicle tracking techniques used to acquire data for travel time estimation and introduces the concept of tracking vehicles based on simulation. Furthermore the iTransIT ITS will be detailed and finally different traffic simulation models will be detailed and discussed to find the most suitable model for this project.

Chapter 3: System Design - The principal system design of the iTranSIM system will be presented including the data model. In addition to that specific design decisions will be discussed and explained.

Chapter 4: Implementation - In this chapter general implementation issues are discussed. Furthermore specific issues encountered during the implementation of the proof-of-concept are illustrated.

Chapter 5: Evaluation - The analysis of the proof-of-concept implementation of the iTranSIM system can be found in this chapter. Furthermore possible evaluation scenarios for an extended iTranSIM implementation are described.

Chapter 6: Conclusions & Future Work - In this chapter the results of the project are summarized. Additionally possible directions for future work are outlined.

2 State of the Art

2.1 Intelligent Transportation Systems

In modern urban environments huge amounts of data accumulate on a daily basis. This can include weather information, traffic and congestion data, information on the public transport or available parking facilities. Traditionally this information is gathered by individual systems, where each system fulfils a specific purpose and collects, stores and utilizes a small set of data relevant for the particular purpose. Such specific applications can include the adaptation of traffic lights or variable speed signs to regulate the traffic flow or displaying available parking spaces for different parking lots in an area. However in general these systems work independently and do not share data or interact in any way. Intelligent transportation systems (ITS) try to overcome this situation by interconnecting different systems, allowing exchange and reuse of data from different sources. The ability to combine data from different sources does not only make for a better working of the underlying systems. If data can be correlated over space or time it can be used by new applications to provide value added services to users.

ITS systems can be divided into different sub groups, depending on the information they process, their purpose or their users [4,5]. For this project Advanced Public Transport Systems (APTS) are the most relevant type of ITS systems. However Advanced Traveller Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS) also play their role in our approach so they will be described briefly as well.

With an ATMS general issues of daily traffic can be addressed. Data about the general road network and traffic conditions is gathered and used to control the traffic flow. An example application of an ATMS is to control and synchronise traffic lights according to the vehicle counts on the incoming links so that the throughput on links with high traffic volumes can be maximised.

The ATIS provides travellers with information that help planning a certain trip. E.g. the ATIS can provide alternative routes to a destination when the normal route is blocked due to road works or an accident. Another possible application for an ATIS could involve finding a suitable parking space based on the availability of spaces and the traffic on the way there. In a next step this system could even allow booking a parking space to ensure its availability on arrival.

An APTS provides the passengers with information on trips and vehicles of public transport providers. This includes travel and arrival times at bus stops and information on possible connections one can take at a certain stop including a prediction of the time available to change to the next vehicle. Another typical system is a multi modal traveller information system which enables the user to plan a trip using all kinds of available transportation. Proof-of-concept implementations of such systems, like the Smart Traveller Information System (STIS), prove the feasibility of this approach [6]. The STIS allows planning a trip in Dublin city and was implemented based on the iTransIT framework which is described in section 2.1.5.

The iTranSIM project intends to provide information about the state of public transport, therefore the next chapter will give a more detailed insight into APTS systems.

2.1.1 Advanced Public Transport Systems

The purpose of APTS systems is to inform passengers about the current and possible future state of the vehicles in the system [7]. This includes pre-trip information as well as on-trip information. Pre-trip information such as start and travel times as well as possible connections and transfer times allow travellers to plan their journey in advanced. On-Trip information is provided for the passengers while they are travelling in the vehicle, detailing e.g. arrival times and possible delays.

The information can be provided to the customers either in-home (by phone or internet), in-terminal or in-vehicle. In-home systems provide information to the travellers before they started their trip, i.e. when they are at home, planning their trip. In-terminal systems provide a similar functionality so passengers can plan their journey when they arrived at the stop. They can also provide updated information for people who planned their journey in advanced or for those just waiting for a particular service. Once boarded, passengers can get up to date information from in-vehicle systems (on-trip information).

With the advent of ubiquitous computing these separate categories start to mix because technologies such as the third generation mobile phone standard (3G) allow access to the internet and the information provided at any time in many different places.

In its most basic form, an APTS uses static data such as timetables and provides it through different channels like the internet or electronic billboards at the stops. More sophisticated systems incorporate live data into the information provided to offer a realistic view on the current situation. The main sources of live information are automatic vehicle location (AVL)

systems. These systems provide up to date information on vehicle locations which in turn can be used to estimate travel times or provide other services based on vehicle location. For example buses or trams can be given a priority at junctions over private traffic or vehicles can be advised to wait a certain amount of time to allow passengers coming in on a delayed connection service to reach their next vehicle. Depending on the variation of travel times these systems significantly increase the service level for passengers. In environments with a large variability of travel times (e.g. caused by traffic) these information and services allow a traveller to move along in a more efficient way. However the main source for these services is a reliable AVL system. As no such system exists at the moment for the buses in Dublin, we will describe and compare different AVL systems in the following section.

2.1.2 Automatic Vehicle Location

The AVL system can locate vehicles e.g. in an urban environment. This data is then transferred to a central server and stored for all vehicles in the system. Depending on the type of vehicle different means of localization and communication can be used.

In 1977 Riter and McCoy gave an overview over automatic vehicle location [8]. At this time the main purpose was not passenger information but to help law enforcement agencies to dispatch their vehicles. However they also mentioned public transport and the possibility to use AVL data to reschedule vehicles and to help operators keeping to their time tables. They identified three major categories which still apply to today's AVL systems:

- dead-reckoning methods
- proximity systems
- radiolocation methods

In addition to these methods we will introduce a new system based on statistics and traffic simulation.

There is a substantial difference in the way AVL can be applied to free moving vehicles such as buses on one hand and track bound vehicles such as trains or trams on the other. In general it can be said that locating trains or trams is easier because their tracks and movement patterns are predetermined whereas a bus can move freely and might leave its predetermined track e.g. when a diversion due to road works is in place along the route.

Dead-Reckoning

In dead-reckoning the track of a vehicle is recorded, i.e. the direction and distance of a trip are stored. When the data is recorded in a high resolution (i.e. several samples per second) the track of a vehicle can be plotted and its position can be determined. The advantage of dead reckoning is that as opposed to the other methods no radio communication equipment is necessary for the localization of vehicles.

The main problem with dead-reckoning is that errors do accumulate because the position calculated depends on previous measurements. External influences such as side winds and internal imprecision in measuring the distance and direction lead to an unacceptable inaccuracy when tracking freely moving vehicles. Track bound vehicles on the other hand have the advantage that the direction of travel is defined by the track system, so only the distance travelled needs to be recorded. Furthermore the position can be adjusted when a train or a tram stop at a station so the error can only accumulate over the distance between two stops. Therefore this system is still in use for tracking track bound vehicles but not for freely moving vehicles. It is also used in combination with other AVL systems to provide a backup when the other systems fail. One example was described by Zaho et. al [9] where GPS data (se below) is enhanced with dead reckoning data to overcome issues with GPS signal reception in build up areas. The Q-Time system, that was deployed in Dublin as a prototype for RTPI (see section 2.1.4), also uses an odometer on the vehicles in addition to other technologies.

Proximity Systems

Proximity systems use receivers in strategically placed locations along the possible vehicle tracks to locate vehicles. Vehicles carry some form of beacon which identifies the vehicle with the receiver. An advantage of proximity systems is that the actual position of the vehicle is captured in the receiver, not in the vehicle. Therefore transferring the data to a central server can be accomplished without the need for wireless communication (for details see below). However to get a fine grained resolution of vehicle locations the number of receivers has to be very high, especially in large urban environments.

Again proximity systems work better on track bound vehicles than on freely moving ones. Tram or train tracks can be equipped with induction loops or light barriers / infra red sensors to locate vehicles on the track. As these vehicles are not supposed to be hampered by other traffic, their location can be calculated using average speeds between the receivers.

Furthermore the distance between receivers can be larger than in road networks for the same reason. Thus proximity systems are mainly used to locate track bound vehicles. Innovative receivers help to build more accurate and cheaper proximity systems. One example of these new receivers are video cameras. Using computer vision, the footage of these cameras can be analysed, recognising license plates and therefore locating the vehicles [10].

Radiolocation

The most common method to locate vehicles nowadays is radiolocation. With this technique the vehicle receives radio signals from different stations and calculates its position based on the delay of the different signals. Thus the vehicle must receive signals from at least 3 stations to triangulate its position and the locations of these stations must be known. Compared to the previously described methods radiolocation allows a more precise location with accuracy in the range of a few meters.

The stations needed for radiolocation can be situated in the environment or satellites can be used. Currently the NAVSTAR global positioning system (GPS) is the main source of signals for radiolocation not only in AVL but for all kinds of localization purposes in the western world. NAVSTAR GPS consists of about 30 satellites and is controlled by the United States department of defence. A European, non military alternative is currently set up under the name of Galileo. Satellite systems such as NAVSTAR GPS or Galileo are superior to radiolocation stations on the ground because they can be used free of charge where ground stations would have to be purpose built, set up and maintained. Therefore satellite based radiolocation has replaced most of the other AVL systems as the main source of vehicle locations. The RTPI system Q-Time that was tested in Dublin (see section 2.1.4) also uses GPS as the main source of vehicle location.

Mobile phone technologies such as the global system for mobile communication (GSM) can also be used as source for radiolocation. The GSM can be used as a proximity system because every mobile device using GSM is logged into a specific cell. A more precise localization of devices is possible using the signal strength of the device's communication with several base stations to triangulate the position. Other approaches combine the GSM data with data from the digital audio broadcast (DAB) to calculate an accurate [11].

Communication

Using either of the technologies described above the location data is generated locally either in the vehicle itself (dead-reckoning, radiolocation) or in some receiver stations (proximity system). This data has to be transferred to a central server so it can be analysed and utilised.

The base stations of proximity systems can be connected using a wired link, provided an appropriate communication infrastructure is already in place. However data gathered in the vehicles has to be communicated to a central server over a wireless link. Usually this is accomplished via the existing radio communication equipment in the vehicles. These systems were introduced to facilitate voice communication between the driver and the control centre. However the radio equipment can also be used for data transfer but might become a bottleneck when too many vehicles try to communicate over a specific link. To ensure collision free communication on a channel the control centre polls all vehicles one after another to receive their locations. Currently polling cycles of 20 to 40 seconds are common for medium sized public transport providers, the Q-Time prototype deployed in Dublin has a polling frequency of 30 seconds (see below, [12]). With a larger fleet the time needed to poll all vehicles increases which has a severe impact on the precision of these systems: although a GPS receiver can update a vehicles position every second the data can only be transmitted once every minute therefore rendering most of the data collected useless.

2.1.3 AVL via Traffic Simulation

AVL systems are a necessity for modern APTS systems and with the availability of GPS the cost of updating a fleet with AVL equipment has come down significantly. However especially for large fleets such as that of Dublin Bus with almost 1100 buses [1] two major issues arise when planning on introducing an AVL system covering the entire fleet. Although satellite based radiolocation has brought down the cost of equipping every vehicle compared to setting up a complete radiolocation or proximity system, the cost of equipping 1100 vehicles with receivers as well as the cost of maintenance is still substantial. The second problem was already described in the previous section: the communication of location data can become the bottleneck when using AVL on a large fleet.

To overcome these problems a different technology for locating vehicles in an urban environment based on online traffic simulation will be proposed in this paper. The main idea behind this concept is that due to availability of ATMS and ATIS systems a large variety of data about the state of traffic in the network is actually available. Furthermore traffic

simulators allow reproducing all kinds of traffic situations in a realistic fashion and simulations can usually run in a multiple of real time. Therefore it should be possible to use such a simulator to recreate the current traffic situation in the simulator, having the simulation running in 'sync' with the real world and being an almost precise copy of the real world. Once the traffic in a city can be rebuilt precisely in a simulation, particular vehicles can be tracked in the simulation as opposed to the real world. Therefore vehicles do not need to be equipped with special receivers and the location data does not need to be communicated over a large distance because it is contained in the simulation. Thus the data can be updated in shorter cycles compared to existing AVL solutions. To the knowledge of the author this is the first time such an approach is actually proposed as an alternative to existing AVL systems.

To realise such an AVL system two major components are needed: the ITS system which can supply the data necessary to create a realistic online simulation and the traffic simulator. The ITS system currently in use in Dublin is the iTransIT system which will be explained in the section 2.1.5. In section 2.2 different traffic simulation models and simulators are described.

2.1.4 APTS and AVL in Dublin

In 2001 a pilot project started, providing real time passenger information for one of Dublin Bus' quality bus corridors (QBC). Now a fleet wide deployment of AVL is planned. Both the pilot project and the new plans are described in this section.

Q-Time

To test the public perception of real time passenger information systems Dublin Bus rolled out a pilot system along the QBC leading from Dublin to Lucan in June 2001. The system consist of electronic displays and buses supplied with an AVL system based on GPS, a wheel pedometer and door sensors [12]. The displays are depicted in Figure 1.

Figure 1, Q-Time - Electronic Display [12]

The vehicles travelling on the Lucan QBC update their position using GPS every 20-30 seconds, the location information is then transmitted to a central server. The server calculates the arrival times and updates the displays at the bus stops.

Studies have shown that the system was very well accepted by the passengers travelling from bus stops equipped with electronic displays [12]. This 3 year pilot project was seen as a proof that real-time passenger information helps to raise acceptance of public transport. Furthermore the project provided useful information for next generation (i.e. fleet wide) information systems in Dublin.

Fleet-wide AVL

Dublin bus has realised that an AVL system is necessary to support their operations and to improve passenger information as can be seen from this excerpt from their bus network review in 2006:

> *Even headways and general reliability have proved difficult to establish in practice, and would require the support of an* ***Automatic Vehicle Location*** *(AVL) system to achieve significant improvements. [13], p. iii*

According to an article in the Irish Independent from April 2006 Dublin Bus is planning to equip all its vehicles with a GPS based AVL system [14]. According to that article the budget for this project is about €10m - €12m. Once started the update of the entire fleet is supposed to take 18 month to 2 years.

Although this would mean that the simulation based AVL is basically unnecessary there are still reasons to conduct this research. Even though the rollout of the AVL system was announced in 2006 there is no updated information available on that manner. Apart from the article there is only a short note in a press release from Dublin Bus mentioning the rollout of this system [15]. It is neither mentioned in the bus network report nor in the 2006 annual report [1, 16]. This leads us to belief that the deployment of such a system does not have a high priority for Dublin Bus, i.e. if the simulation based AVL yields accurate results it could be deployed instead of the GPS based version. Alternatively it could be used at least during the two years when the AVL system is deployed as the simulation based approach would provide results immediately. Once the other system is ready it could then replace the simulation based one.

2.1.5 iTransIT

When a new ITS system is installed all components could be built so they adhere to the ITS' specifications and are capable of communicating with one another. However in many cases such as in Dublin there are already legacy systems deployed which are purpose built and each system uses its own, proprietary data formats and communication protocols. In that case it is important to ensure the extensibility of the ITS so that different legacy systems can be addressed and combined.

Apart from the ability to include a large variety of systems for data retrieval, ITS systems also need to provide a structured way of accessing this data. Providing standardized means of information access to higher level applications ensures that programmers can easily write new applications to leverage from the data collected and therefore to provide value added services to the user.

The iTransIT ITS was developed by the Distributed Systems Group at the Trinity College Dublin. It is a framework for building ITS systems and for integrating existing legacy systems into one single ITS [16-20].

The iTransIT system uses a layered approach where different tiers provide the optimal integration for systems which adhere to the iTransIT data model, for legacy systems using proprietary data models and infrastructure and for applications which query these systems to provide value added services. The data in the iTransIT system can be accessed by exploiting a topographical model using a spatially enhanced application programming interface (spatial API). The three-tier architecture and the spatial API will be explained in the following sections.

2.1.5.1 Architecture

The iTransIT framework uses a three tier architecture to combine legacy systems, iTransIT components and applications which access the data in the ITS. The architecture is depicted in Figure 2.

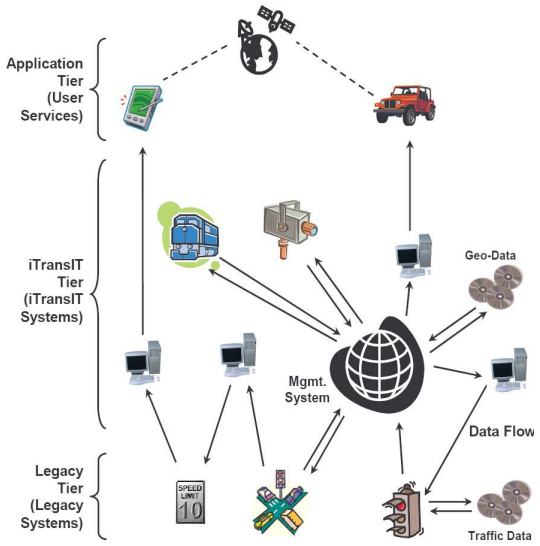

Figure 2, The iTransIT architecture [20]

The legacy tier contains all the ITS systems which are not using the iTransIT data model. This includes components already deployed as well as future systems which might be added into the system at a later stage.

The iTransIT tier exists between the underlying legacy systems and the applications which use the ITS data to create value added services. The tier can contain several iTransIT Systems, the iTransIT Management System and a spatially enhanced database for storing the data gathered by the ITS. The common data model which contains the data available in the ITS is also contained in this layer. However there is not a single entity containing the whole data model but the data is distributed over the existing iTransIT systems. Each iTransIT System fulfils a specific function within the ITS, executing one of the following roles:

- Dedicated User Service: This system acts as a middleware between a legacy system and user services in the application tier. It enables services to use the spatial API to gain their data although the system providing the data (i.e. the legacy system) does not support this data format.
- Legacy System Monitor: Such a system provides facilities for two legacy systems to communicate with each other.
- Universal Processor: This system operates only in the iTransIT layer and provide capabilities to either capture data or perform calculations to support other iTransIT systems.

- Universal User Service: This system combines data form different iTransIT systems to infer higher level knowledge. This new data is then provided to user services in the application tier.
- Dedicated Processor: A dedicated processor acts as a middleware between iTransIT systems and a legacy system. It gathers existing data using the spatial API and converts the data into a format that complies with the legacy system.

The iTransIT Management system is a more generic application which contains the main part of the common data model and constitutes the main access to the spatial data stored in the ITS. It is used both by iTransIT and legacy systems to retrieve and store data.

The application tier sits on top of the iTransIT tier and provides access to the data gathered either by legacy or iTransIT systems. In this tier value added services are implemented which access the data using the spatial API and provide information to the user.

This architecture provides the facilities to combine not only iTransIT systems but also legacy systems of any kind into one ITS. Therefore the iTransIT architecture can be applied to existing ITS systems and is easily extensible. Data can be exchanged between different systems to create more complex applications based on available data. Furthermore services can be implemented to provide the users with additional information helping them to plan their journeys or to accomplish specific tasks in an urban environment. However the complexity of the underlying systems is hidden to the services in the application tier, only the unified spatial API is used to retrieve the relevant data. This makes it easier for application developers to create value added services because an understanding of the underlying sub systems is not necessary.

2.1.5.2 Spatial Programming Model

Besides the ability to share data between different systems a main feature of the iTransIT framework is to provide developers with a standardised interface to the data gathered by different systems. This way, new systems which rely on this data can be developed and deployed easily. In iTransIT the data is stored in a spatial data model and can be accessed trough a spatially enhanced API (spatial API).

Spatial Objects

The main aspect of the data model used in iTransIT is that all the information is stored as so called spatial objects. All objects are derived form one of the three main objects:

- Real World Object: These objects represent physical entities, and contain a location object describing this entity's position or its dimensions.
- System Object: A system object describes one particular subsystem in the ITS which can be either an iTransIT or a legacy system.
- Data Object: The data gathered by the ITS is stored in data objects. They can be associated with either system or real world objects.

One example for the different objects is a weather recording system. The state of the system (e.g. is the system operational or not, which stations need maintenance) is stored in a system object. Each of the stations connected to the system is represented by a real world object; the data gathered by the station (e.g. amount of rain, temperature) is stored in data objects.

This set of objects was developed in cooperation with the traffic office of the Dublin City Council. This was to guarantee that although it is rather small and coarse grained it allows modelling all aspects crucial to the ITS in Dublin or any other ITS for that matter. The small generic set allows extending the ITS without any changes to the core data model. New systems can be modelled as a combination of objects derived from one of those described above.

Spatial API

As the name suggests the main paradigm behind the spatial API is the usage of spatial context. Objects that are derived from real world objects have a location associated with them, alongside with data objects to store the captured data. Every object has a type and a unique identifier through which they can be addressed for data retrieval or update.

The spatial API comes with a number of methods to retrieve object identifiers for a certain type so another system or service can use the data contained in those objects. The ITS can be queried for objects within a certain location hence exploiting the spatial context given with each real world object. Furthermore the temporal context is also captured in the model as each data object has an attribute storing the time of retrieval. Thus objects with a specific temporal context can also be requested. In addition to the spatial and temporal context certain attributes can be used to filter the retrieved objects.

The following situations are to clarify the methods for object retrieval: If a customer wants to know where the nearest bus stop is located an appropriate service can take the user's current location and query the ITS for bus stops in close proximity (spatial context: location of bus stops). Furthermore the user might specify a particular route he likes to use, thus the iTransIT system can be queried for bus stops in close proximity which are served by this route (spatial context: stop location, attributes: stop served by a specific bus route). When travel times are to be calculated for a certain bus its position and the time when the position was recorded are needed to compute the time it will take the vehicle to arrive at the bus stop (attribute: vehicle operating on a specific route, spatial context: location of the vehicle, temporal context: time of location retrieval).

In addition to data retrieval the objects can also be updated via the spatial API. After the identifier of the object in question was retrieved using the methods described above the attributes of this object can be changed. An AVL system can for example use this method to update the position of a certain vehicle in the iTransIT system.

2.1.6 Conclusion

To sum it up intelligent transportation systems allow sharing the data of existing traffic management, traveller information and public transport systems. From the accumulated data new knowledge can be inferred and used to provide value added services. One of these services is a passenger information service for travellers on public transport in urban environments, allowing them to plan their journeys.

The main source of data for an APTS is the automatic vehicle location system which constantly retrieves the position of the public transport vehicles. The three main types of AVL systems were introduced: dead-reckoning, proximity and radiolocation systems. However equipping a large fleet such as the one of Dublin Bus with either of those AVL systems imposes a big financial burden on the operator. Furthermore technical problems such as the timely communication of the results to a central server can either limit the capabilities of the system or significantly increase its operating costs.

To overcome the problems associated with existing AVL solutions, a new method of locating vehicles will be explored in this project. Instead of equipping vehicles with tracking technology an online simulation of the traffic system is used to locate vehicles based on statistical data. If the data needed can be provided by an existing ITS system the cost of rolling out such an AVL system is only a fraction of the cost for traditional AVL.

The iTransIT ITS system in Dublin would be capable of providing the information needed to create an online simulation. It also provides a data model which supports the modelling of an AVL system based on simulation and a programming interface that offers a unified access to the data. Therefore the iTransIT system provides an appropriate foundation for this project.

2.2 Simulation

In the day to day running of traffic systems, especially in urban environments, it is important to minimize the congestion of the road network by optimizing the throughput. The optimization can be carried out using field research and field experiments, but this approach is rather limited: a new timing for a traffic light can be tested; however measuring the effects of replacing a traffic light junction with a roundabout a priori is impossible. Thus there is a need for simulating the traffic flow which allows trying different setups in a safe environment. Furthermore this approach allows a faster evaluation of the proposed changes as simulations can be executed faster than real time, provided the appropriate computational power is available.

2.2.1 Models

The main method of vehicular traffic modelling is creating mathematical models which allow describing the physical propagation of traffic [21]. The research in this field started with the first attempts by Lighthill and Whitham [22] to describe vehicles in traffic flows as particles in a fluid, thus allowing them to use this well studied area of physics to calculate traffic flows. From that point onwards the topic has been actively discussed in the scientific community, bringing forward many different models which can be categorised in different ways. A key variable for distinguishing different classes of traffic flow models is the level of detail [23] which is also used in the following section giving a brief overview of the topic.

There are two fundamental principles when it comes to modelling traffic flows: macroscopic and microscopic models. Macroscopic models try to model traffic flow as a single entity using the analogy to continuum mechanics and hydrodynamics. Microscopic models on the other hand model every single vehicle with respect to the other vehicles around it (car following model). In between there are the mesoscopic models which model some parts as a whole (e.g. the traffic flow) whereas other aspects are modelled for each individual vehicle (e.g. the traffic flow is modelled as a continuum but specific actions such as lane changes are modelled individually for each vehicle). A last category are the sub-microscopic models which do not only take the vehicles in the surroundings into account but also the factors of the vehicles inner workings, including the driver. For example the driver's behaviour can be modelled in sub-microscopic models using expertise form psychology. There are also several approaches to combine different models to create more specialised models which fit a specific purpose.

As the main purpose of this thesis is to prove the feasibility of using traffic simulations for tracking public transport vehicles, the microscopic model is detailed below as it allows modelling the vehicle (e.g. a bus) and thus tracking it individually. In the remainder of this section different flow models and their implementations will be explained.

Microscopic Traffic Flow Models

In microscopic flow models the main idea is that the driver of a vehicle adapts his behaviour according to the vehicle in front of him. The early models are built upon a simple car following schema based on the safety distance. There is a minimum distance between two vehicles and this distance depends on the velocity of the leading car. This model was later refined to include the reaction time needed by a driver to respond to acceleration / deceleration of the leading vehicle. Instead of the safety distance the stimulus-response model describes the driver's behaviour as a response to the behaviour of the leading car. This approach is very static, i.e. a driver's behaviour is influenced even if there is a big gap between the two vehicles and there is no differentiation between different kinds of vehicles / drivers, e.g. faster personal cars v. slower trucks. Thus this model was enhanced introducing limitations on the driver's reactions on stimuli. This model is called the psycho-spacing model and was first developed by Wiedemann in 1974. He also included lane changes and overtaking into his model; previously only simpler single lane traffic was modelled. Most of the recently proposed microscopic traffic flow models are based on the psycho spacing model [23].

Microscopic traffic simulators are generally discrete in time and continuous in space. The position of a vehicle is determined on a continuous basis (e.g. coordinates or distance travelled on a specific road) for a discrete moment in time [24,25]. Depending on the implementation a timeslot is usually in the order of seconds to milliseconds. Apart from the space continuous models there are also fully discrete models which are discrete in time and space.) One example for such an approach is the model by Esser and Schreckenberg based on cellular automata [26] (see section 2.2.3 for details).

For a more detailed overview over different microscopic traffic models the reader is referred to the results of the SMARTEST project which evaluated 32 different traffic models [27].

2.2.2 INTEGRATION

INTEGRATION originally started out as a model for both freeways and arterials. However in later versions enhancements were made towards the integration of ITS systems and the modelling of vehicle emissions [25]. INTEGRATION is a hybrid system modelling both individual vehicles (microscopic) and the overall macroscopic traffic flow. Thus according to the designers some sources consider it a mesoscopic model, they refer to it both as a mesoscopic and a microscopic model in different parts of their publications.

2.2.2.1 Model

The main components of the model are the fundamental traffic flow model, the car following and lane changing, and dynamic traffic assignment routines. Thereby the following driver decisions are to be replicated: selection of the route (dynamic, might change during the trip), selection of speed / lane and decisions on whether to cross an uncontrolled junction based on the gap to the upcoming vehicles (e.g. right turns without traffic lights or STOP / Give way junctions).

Vehicle Behaviour

On its start the vehicle enters its origin link on the lane with the biggest headway to the next vehicle. The speed is than calculated based on the headway to the next vehicle which is driving in the same lane. Then the distance travelled in one timeslot (one deci-second for the implementation described by Aerde et. al [25]) is calculated for the vehicles and they are placed at their new position. Although the speed is calculated on a microscopic car following method the overall traffic flow is managed using a macroscopic calibration. For a detailed description the reader is referred to Aerde & Rakha [28].

Lane Changing

The other main part of the INTEGRATION model is the lane changing functionality. Lane changes can either be discretionary or mandatory. Discretionary lane changes can be performed when the change promises a better progress the other link, e.g. because the shoulder lane is busier than the middle lane so on the middle lane a higher speed can be reached. Mandatory lane changes describe changes necessary to enter the next link on the trip, e.g. an off ramp. Lane changes are subject to a gap large enough permitting the change. If there is no appropriate gap for a discretionary change the car will stay in its lane and

continue travelling, if there is not enough space for a mandatory change then the vehicle will slow down and ultimately stop to wait for a gap that permits a lane change.

2.2.2.2 Implementation

Aerde et al. also described an implementation of the INTEGRTION model [25]. The main features of that implementation are described in the next section

Road Network

The road network in the INTEGRATION model is modelled as a set of links, which are allowing traffic to flow in one direction, on one or more lanes. At the end of each link (e.g. a junction) the vehicle will enter the next link until it reaches its final destination. The links can also be restricted in different ways. Lanes can be restricted for certain kinds of vehicles (e.g. bus lanes). This allows defining specific lanes within a link which are either to be used exclusively by a specific type of vehicle or from which a specific vehicle type is excluded. Furthermore turning movements can be restricted: they can be either bound to specific lanes (exclusive lanes) or they can be banned, i.e. they are not permitted although the road geometry would allow them.

The model separates two main types of links: signalled and unsignalled links (freeways). For the freeway sections merges, diverges and weaving can be modelled. For each of these structures queuing can occur and will be handled appropriately by the model's logic. Signalled links are somewhat equal to freeway links but exit privileges are suspended for certain time periods to simulate red traffic lights. In the model this is realised by creating an extra vehicle on this link and positioning it right on the junction. As the vehicles adapt their speed according to the distance to the vehicle in front of them (as described below) they will ultimately stop at the red traffic light. Furthermore a microscopic gap acceptance schema is implemented in the INTEGRATION model to model either permissive right or left turns on red or stop / yield signs. The model's logic determines from the flow rate whether or not a vehicle can cross the opposing traffic flow.

Routing

Vehicle trips in the INTEGRATION model are specified based on origin destination information. Rather than having vehicles performing random turns at the end of links the route is determined by a start point (origin), an end point (destination) and a routing table.

The routes in these tables can be static and deterministic, i.e. It is either predetermined (e.g. by the modeller or automatically, e.g. using Dijkstra's algorithm) or they can be dynamic and stochastic, i.e. determined based on the current congestion levels on a link. The INTEGRATION model allows the coexistence of several routing tables e.g. to facilitate multi-path traffic assignment.

Additional Features

A last feature to be mentioned here is the simulation of incidents. Incidents can basically be modelled at any point in time, for a specific duration, on certain lanes of a link. These lanes can be blocked completely or the throughput can be limited to a certain percentage of he original value (e.g. grass cutting works can be modelled as a 25% decrease of throughput as they are not completely blocking the whole link.) The dynamic routing reacts in a way that once the incident has happened, it will react on longer delays on this link and find a diversion which allows a faster trip. Thus the flow does not change until the incident is actually happening.

Output

To capture the data from the simulation several methods are provided. Link travel times are measured for every vehicle on a link and a weighted average is used to represent the average travel time for a link. Induction loops can be inserted into the road network to record vehicle counts and speeds. Furthermore these detectors can be limited to certain vehicle types, e.g. to get measures specific to public or private transport only. Vehicle probes are also implemented, allowing different levels of detail. The simplest probe just reports the start or the end of a journey, the most complex generates a report every second to inform about vehicle speed, number of stops, etc.

2.2.2.3 Summary

Summing it up the INTEGRATION model for traffic flow modelling is a very advanced model as it allows for the specification and simulation of different types of road networks (freeways and arterials). It is also unique in the way that it includes macroscopic features into the microscopic model. Some of the features could proof useful when modelling public transport, e.g. the exclusive bus lanes which exist in Dublin City could be modelled easily as this is a feature of the INTEGRATION model. Although the model was implemented several times, [25, 29] there are no implementations freely available at the moment. There are

detailed descriptions of the model available but implementing it from scratch is not feasible for this thesis as its main purpose is to generally proof the feasibility of using traffic simulators to gather live information on public transport vehicles. However this model could be investigated further once the general concept is verified.

2.2.3 Cellular Automata

Cellular Automata allow modelling traffic flow in a time and space discrete manner. They are based on the Nagel-Schreckenberg model [26]. The model and existing implementations are explained in this section.

2.2.3.1 Model

Where other microscopic models use a float value to describe the position of a vehicle in the system (either 2D coordinates or the distance travelled on a specific link) the cellular automata approach splits a lane into several cells. These cells have a specific size (e.g. 7.5m in the simulator of Esser and Schreckenberg [26]) and are either occupied by a vehicle or they are free. The speed of a vehicle is not assumed to be in $km \cdot h^{-1}$ or $m \cdot s^{-1}$ but it represents the amount of cells a vehicle can travel in one timeslot (timeslot is set to be one second in the simulator of Esser and Schreckenberg [26]).

Vehicle Behaviour

The system is updated according to production rules, determining the new speed of a vehicle based on the gap to the preceding vehicle. Furthermore a randomization is included to simulate an imperfect behaviour of a driver. Following are the rules used to update a vehicles position [26]:

1. Acceleration: $v \rightarrow \min(v + 1; v_{max})$
2. Avoiding crashes: $v \rightarrow \min(v; gap)$
3. Randomization: if $rand() < p_{dec}$ then $v \rightarrow \max(v - 1; 0)$
4. Update: Each vehicle is advanced v cells.

In these rules v denotes the current speed (cells/timeslot), gap is the number of free cells between the vehicle and the one in front of it, p_{dec} describes the probability for deceleration despite a free track.

As this model was originally intended for single lane roads lane changing behaviour was added to the simulator as well as other extensions to create a more realistic model.

Lane Changing

For the most part of an edge lane changes are unrestricted, however coming towards the end of an edge a vehicle enters the DirectionChange zone which allows lane changes only when they are necessary to reach the destination (e.g. entering an off-ramp). Adjacent to the DirectionChange zone there is the ChangeLock zone. This zone spans to the end of an edge and does not permit lane changes. If a vehicle reaches the end of the DirectionChange zone but is not in a lane which allows travelling in the desired direction the vehicle has to wait until the appropriate lane is free so it can perform a lane change before entering the ChangeLock zone. The model of a multi lane edge with its different zones is depicted in Figure 3.

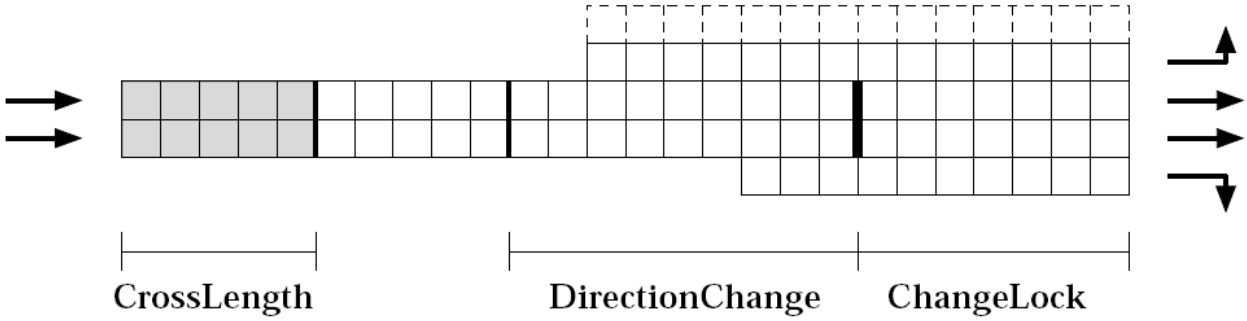

Figure 3, Multiple Lanes in Cellular Automata Network [26]

To model on-ramps as usually found on freeways, there is a transfer edge which allows two separate edges to merge into one.

2.2.3.2 Implementation

Implementations of microscopic traffic simulation based on cellular automata were successfully used, e.g. to model the traffic in the city of Duisburg, Germany [26, 30]. The main features of the implementation are described in the next section.

Road Network

The road network is modelled as a combination of roads and crossings. In accordance with established graph theory crossings are also referred to as nodes and roads as edges. An edge can consist of one ore more lanes allowing travel in the same direction. On single lane edges

the standard cellular automata model is implemented, i.e. no lane changing is necessary. On multi lane edges the lane changing behaviour varies depending on the position of the vehicle.

Nodes (junctions) are points where several edges join. At the end an edge has a leave flag which can be switched to permit or refuse a vehicle to exit the lane. The switching occurs according to timetables which need to be set for each individual traffic light. To keep complexity low only green and red traffic lights can be modelled (vehicles are either allowed to exit an edge or they are not). In addition to the traffic lights priority rules can be set up to model situations where one vehicle has to give way to an other one (e.g. a vehicle crossing the opposite lane with upcoming traffic). To allow more realistic modelling beyond traffic lights and priority rules probabilities can be used to reduce the throughput of a junction. This can be used to simulate pedestrian crossings or other impacts on the network which force the vehicle to stop although it has a green traffic light and/or priority on the lane.

Routing

The Simulator of Esser and Schreckenberg allows vehicles to move around in random fashion or according to a specific route. With a random movement a vehicle picks a direction to go before it actually reaches a junction so it can perform the appropriate lane changes prior to reaching the ChangeLock zone. Routes can also be predetermined for certain vehicles, i.e. for every junction the direction to go is predefined. Different vehicles can use different strategies in the same simulation, however one vehicle can only be guided by either random or route based movement.

Vehicles enter the simulation at sources and leave at sinks. Because vehicles can join or exit the traffic at arbitrary positions in an urban environment [26] every cell in their simulator can be defined to be a source or a sink. Furthermore a combination of a sink and a source (a flow-checkpoint) can be used to adjust the number of vehicles travelling on a certain road. Another combination of a source and a sink on one cell is used to model parking spaces; the cell can be marked as occupied or free.

Additional Features

Different vehicle types can be used in the simulator. All instances of one type have certain properties in common (e.g. maximum speed, length). Special types are predefined to model public transport and blocked roads. A public transport vehicle (e.g. a bus) traverses predefined routes in regular intervals. Additionally stops can be defined where the vehicle

waits to simulate passengers entering or leaving. Blocked roads (e.g. after an accident or during road works) can be simulated using vehicles of a certain length (i.e. the length of the road block) and with a maximum speed of 0.

Output

To acquire information from the simulator different means of measurement are implemented. Measure points are the equivalent of induction loops in the real world. They collect data about the number and the average speed of vehicles on a certain point of the road network. Measure regions do the same but for a larger area, i.e. for one or more edges. Furthermore vehicle probes can be used to track specific vehicles in the network. Vehicle probes acquire data like the travel time, route travelled (important for random movement), number of stops and waiting time at traffic lights.

2.2.3.3 Summary

All together the Cellular Automata Model has most of the features necessary to implement a public transport control system. Especially the flow-checkpoints which allow adapting traffic to real, live vehicle counts is one of the main elements when it comes to producing an online simulation of the real traffic situation. There are facilities to model public transport vehicles and the vehicle probes could be used to extract the relevant information, e.g. the position of a bus, from the simulator. Unfortunately there is no implementation of this model publicly available for testing. Thus, just as with the INTEGRATION model described in section 2.2.2, it would be necessary to implement a full simulator before the test could be run on it. This overhead is not acceptable for this feasibility study.

However as the model uses a discrete space it is also superior to many other models with regards to computational speed. It therefore would scale better to a large network such as the model of a multi million inhabitants urban area like the city of Dublin. Although the Cellular Automata Model will not be used for this proof-of-concept it could very well be used for a future application gaining data on public transport on a city-wide basis.

2.2.4 SUMO (Gipps / Kraus)

SUMO is an open source traffic simulator based on discrete time (time steps) and continuous space. In the SUMO framework the time step is one second long, i.e. a time step equals one second of real world traffic.

2.2.4.1 Model

The model used in SUMO was designed by Gipps [32] and later extended by Kraus [24].

Vehicle Behaviour

The model implements collision free car following. A vehicle's speed is determined by the speed of vehicle in front. A safety gap is held between the vehicles to prevent collisions. In each time step the desired speed is recalculated and adapted to fit the vehicle's physical abilities of acceleration and deceleration. The model also allows modelling imperfect driving behaviour by fluctuating the vehicle's speed.

Lane Changing

As the previously discussed models the Gipps / Kraus model also supports multi lane traffic and lane changing. Just as its car following model the lane changing model is collision free, so lane changes only occur if the gap on the other lane is big enough for vehicle to safely swap lanes. Lane changes are made whenever there is an advantage out of it (i.e. the vehicle in front is too slow and the speed limit is not yet reached). However with a certain (configurable) probability lane changes can also occur without an imminent benefit. This helps modelling the imperfect human behaviour.

2.2.4.2 Implementation

The Gipps/Kraus model is implemented in the SUMO traffic simulation toolkit [33], which is described in this section.

Road Network

The road network in SUMO consists of nodes and edges, just as in the cellular automata simulator described in section 2.2.3. Again an edge connects two nodes and represents one or several lanes going in the same direction. The nodes can be either controlled by a traffic light or by using priorities for the different edges. Each traffic light can be configured individually; if there is no data on the traffic light sequence available it can be generated automatically. Edge priorities can be used as an input into the traffic light sequencer so that the output closely resembles the real traffic light sequence.

Routing

There are several approaches to emitting and routing cars are implemented in SUMO. Generally vehicles are associated with an explicit route and a specific start time. At the given time the vehicle enters the simulation at the start of the route, follows the route and is taken out of the simulation once it reaches its destination. These routes can be built from the following data:

- random data
- trip data
- origin / destination (OD) matrices
- junction turning ratios
- induction loop counts

Random data allows creating the simplest form of routes. These routes are not very reliable and should be used only for testing purposes. The problem with random routes, although they might be valid routes within the network, is that the differentiation between arterials and minor roads is discarded. For the random route generator all routes are equally important. Thus minor roads are used disproportionately often compared to the real world.

Trip data describes a single trip within the network. Information on the trip include origin, destination and start time of a single vehicle. To generate routes based on trip data, very precise information about activities of vehicles in the real world is necessary.

An OD matrix is a generalization of trip data. It contains the same information, but in a more generalised way, i.e. rather than having exact data on origins and destinations, OD matrices usually store the area of origin / destination, to allow for a bigger variety. They are also applied to a number of vehicles, not to a single vehicle only. The information necessary to build OD matrices is vaguer than the data needed for building routes from specific trips. However a general idea about the movement of the population is still needed.

Junction turning ratios are probabilistic values describing the likelihood for a vehicle to exit a junction into a certain direction. Routes that are likely to occur in the real world can be calculated using these ratios. This is possible even without information on the real trips. The junction turning ratios can be obtained e.g. by using induction loops in and around a junction or empirical data. If this data is available information on the individual behaviour of the general public is not needed.

If induction loops are spread over the whole network routes and flows (amount of vehicles in a certain timeslot) can be generated automatically. Just like the junction turning ratio approach, this way of generating routes does not rely on any trip data. The advantage over the junction turning ratio router is that not only routes but also the amount of vehicles can be determined using network-wide induction loops.

All the methods described above produce static routes, i.e. a sequence of kinks which are used to get from the starting link to the end. A dynamic routing e.g. based on junction turning ratios (for each vehicle the next link is calculated 'on the fly' during the simulation) is not possible with SUMO, however re-routing of vehicles is possible. In this case the route taken by a vehicle can be exchanged for another route during the simulation.

Once the routes are calculated vehicles can either be put into the network explicitly or they can start at emitters. If vehicles are started specifically a route and start time must be specified. This can be useful if specific information on certain vehicles (e.g. buses in public transport) is available. Emitters are special sources which can be placed in within the network. They can emit single vehicles or a flow (a vehicle every *n* time steps). Emitters in SUMO also allow for a probabilistic choice of vehicle types (see below) and routes. I.e. an emitter can be configured so it emits e.g. a small car in 70% and a big car in 30% of the cases, each of which will choose route A with a probability of 90% and route B with 10%.

Additional Features

SUMO supports different vehicle types. This allows for modelling vehicles with different physical properties (acceleration, deceleration, length, maximum speed) as well as giving different privileges to certain vehicles. Lanes can be configured so they can either be used only by certain vehicles or certain vehicles are prohibited using the lane. The former allows for modelling e.g. bus lanes, latter to implement HGV bans which exist on certain roads.

There are a few more features available in SUMO, e.g. public transport can be modelled explicitly, i.e. the routes, stops and the stop durations can be used to generate a model of a public transport vehicle such as a bus in the simulator. Furthermore there are features to allow a more dynamic simulation of the real world: variable traffic signs allow adapting the maximum speed on a road or a lane, dynamic traffic light management can be used to fine tune the traffic light sequence to the traffic.

For the simulation SUMO offers two programs: a command line tool and a graphical version. Both versions take the same input files and can create the same output files. The graphical

version displays the network and allows tracking the vehicles visually. This is an overhead and not necessarily needed when statistical results are required. The command line version offers an alternative that allows a faster simulation than the graphical version. This can proof useful when information such as vehicle positions or congestion levels are to be generated. Especially for large networks the non-graphical simulation computes much faster than its graphical counterpart.

Output

To acquire data from the simulation SUMO offers several kinds of detectors which can be placed in the network. Virtual induction loops can be used to gather vehicle counts and average speeds over an arbitrary amount of time. They can be placed on any lane and at any position on this lane. Lane based detectors are used to aggregate data over a complete lane. The data gathered herein is the same as in the induction loops. A multi-origin / multi-destination detector allows data gathering over arbitrary parts of the network.

Global data can be collected in addition to the detector data. The network state dump returns the overall data of the network at every time step. This includes information on each lane, including which vehicles are occupying it, how long they spent on this lane, how often they had to stop and other related data. The vehicle emission state dump and travel time dump contain amongst other data global information on how many vehicles are in the network, how many were emitted and how many are waiting. Furthermore average travel times aggregated for all vehicles in the network are collected. Global data on vehicles is also available: vehicle trip information is generated once a vehicle has finished its trip containing start and end time plus further data such as waiting times for each vehicle trip. More detailed route information for every vehicle can also be generated.

2.2.4.3 Summary

Altogether SUMO has many features which make it a good candidate for this project. It allows modelling of urban environments, exclusive lanes for public transport, automatic generation of traffic light sequences. Furthermore the routing based on the junction turning ratios is a good tool to generate realistic traffic without information on the actual travel behaviour of the population (i.e. on specific trips or at least rough estimates of origins and destinations). The main drawback with SUMO is that, unlike in the simulator based on cellular automata, it is currently not possible to dynamically adapt the traffic flows in a simulation. However this is necessary to adjust the simulation to the real traffic. Despite this

shortcoming a big advantage of SUMO is that it is open source software and available under the GNU general public license (GPL). This means that there is full access to the source code which allows for implementing the missing features. Another advantage is that in SUMO all input and output files are in a well documented XML format. This makes the data human readable and allows for processing by other programs.

2.2.5 Conclusion

To generate an online simulation that is capable of providing vehicle locations, a microscopic traffic simulation is needed because in microscopic traffic models every vehicle is modelled individually and therefore traceable.

All of the models described above could be used to realise online simulations and to track vehicles therein. The decision to use the Gibbs / Kraus model was made because a working implementation of this model is freely available. Although there are not all features available in this implementation the amount of work necessary to adapt it to fit the purpose is still smaller than what it would take to implement a complete simulator. This leaves us more time to evaluate the project. Furthermore data on the accuracy and calibration of this simulator already exists and can be used in this project.

3 Design

To use a traffic simulation for vehicle location requires a model of the public transport vehicle's journey within its urban environment. The data from the model then needs to be fed into the simulation. The following sections will describe the design of an iTransIT system which handles the translation between the data model and the simulation. Furthermore each part of the model is described in detail. As the traffic in the simulation has to be calibrated to match the live data the last section of this chapter deals with the design of such a calibrator.

3.1 System Design

To comply with the existing iTransIT architecture our system will be split up into two subsystems. One is a dedicated processor (iTranSIM-DP), the other one is a dedicated user service (iTranSIM-DUS). The model is depicted in Figure 4.

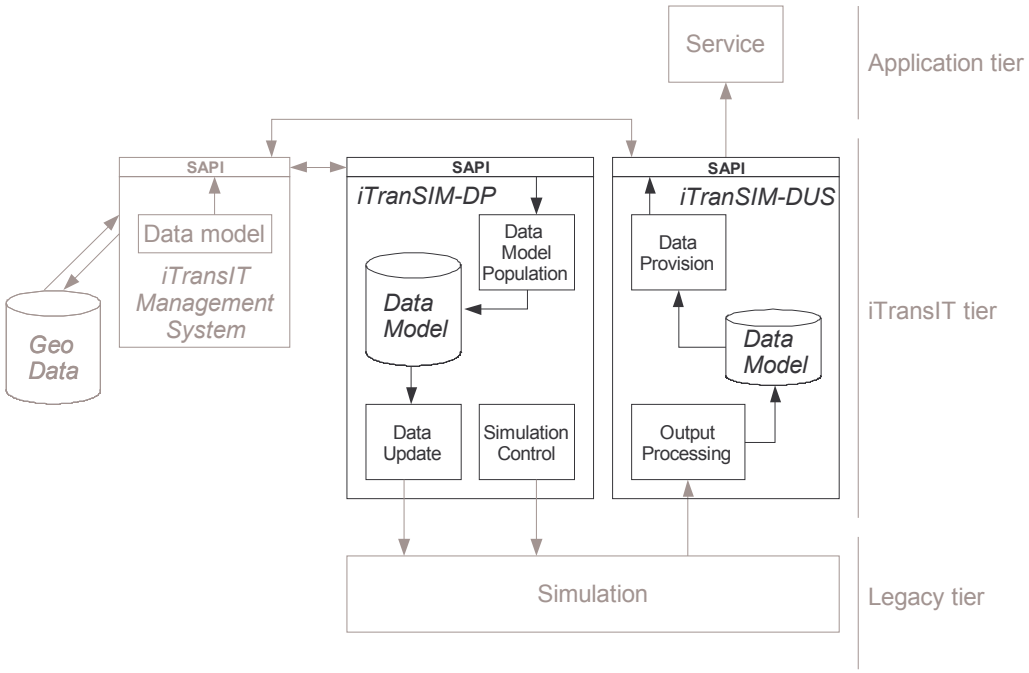

Figure 4, iTranSIM system design

The first system acts as a dedicated processor (iTranSIM-DP) to supply the simulation with all the data necessary to create an online simulation. It contains three main components: One retrieves the data to fill the data model, one converts the data stored in the data model into a legacy format which is supported by the simulation and the last component controls the simulation to ensure proper functioning and timeliness.

A second iTransIT system will act as a dedicated user service (iTranSIM-DUS), processing the output of the simulation and providing it to services in the application tier.

The data model in the iTranSIM system can be seen as an extension to the existing data model implemented in the iTransIT management system which can be accessed through the spatial API.

3.2 Data Model

There are different aspects to the model which need to be considered. First there is the road network which describes the layout of the streets and junctions. We regard the behaviour of public transport vehicles a special form of the general behaviour of private traffic. Thus the second section which needs to be investigated is the behaviour of cars on the road network. Aspects of this model include acceleration and deceleration, influence of regulated and unregulated junctions, the impact of other vehicles within the network. The last component to constitute a bus journey is the modelling of those aspects that distinguish public from private transport, namely the usage of specific routes at specific times and additional stops during which passengers can enter or leave the vehicle. The following sections will describe each part of the model in detail.

3.2.1 Road Network

The road network is essential as it represents the available infrastructure of the environment in question. Especially for urban areas this network can be very complex and it is therefore vital to reproduce it accurately. The road network is important for modelling the journeys of public and private transport in several ways. The density of traffic on a certain strip of road depends on the capacity of the road, i.e. how many lanes are available. The routes vehicles take rely upon the legal turning manoeuvres they can perform at any given junction. Travel times are influenced by the length of the route taken for the specific trip, by the traffic on this particular route and by waiting times on junctions (caused by traffic light sequences or traffic with higher priority).

3.2.1.1 Components

The road network can be represented as a directed graph where the junctions are represented by the nodes and the roads connecting the junctions are the edges of the graph. As some roads may only allow traffic flow in one direction it is necessary to use a directed graph, although this implies the overhead of having two edges for an ordinary strip of road (i.e. one per direction). For this reason the connection between two junctions is from now on referred to as

a link, with a road consisting of one or two links, the former allowing travel in one direction only, the later allowing travel in both directions.

Junctions

Relevant to creating the nodes in the graph is the position of the junction with respect to some reference point in a coordinate system. As the model is to represent a road network in an urban environment, the map can be represented as a two dimensional Cartesian coordinate system. Thus a junction is a point with an X and a Y coordinate, a link consists of at least two points connecting two junctions.

The order in which vehicles can cross a junction depends on the junction type. The type can be one of the following:

- Priority based
- Traffic light controlled
- Roundabout
- On- / Off-ramp

For priority based junctions a vehicle on an incoming link with the lower priority will give way to vehicles on higher priority incoming links. If two links have the same priority the right before left rule will take effect, i.e. if a vehicle approaches from the right it gets to cross the junction first. For traffic light junctions the sequence of the traffic light is important as not all incoming links get the same amount of free flow time (i.e. the green light). For roundabouts no additional information is necessary as the vehicle in the roundabout has priority over vehicles on incoming links. For on- / off-ramps the length of the ramp is important because it determines how much a vehicle can accelerate on the ramp or how many vehicles can queue in case of traffic jams. However ramps are more common on freeways than in urban environments. In fact there are no ramp junctions in the inner city of Dublin. Therefore information specific to on- / off-ramps are not incorporated into this model.

Links

To create a realistic model of a link it can also contain more points which are regarded as waypoints. These points build a polyline from which the exact shape of a link as well as its length can be computed. The length is needed to calculate the time a vehicle will spend

travelling along this link. Links also have a speed limit to constrain vehicles to a maximum velocity while moving along.

A link consists of at least one lane. Each lane allows vehicles moving along the link; vehicles can change the lane they use if there is a benefit from it (e.g. overtaking a slower car). Links can also be restricted to certain vehicles types, e.g. to public transport vehicles. Vehicle types can also be excluded from using certain links, e.g. HGVs are not allowed on any but the outer lane of a link to keep congestion due to these vehicles overtaking one another low. Furthermore links can contain induction loops, counting the number of vehicles passing over them in a specific time period. For the modelling of the road network solely the position of the loops is relevant. The vehicle counts are then needed for modelling the traffic on the network (see section 3.2.2).

The model presented so far is rather static as it contains only data which does not change on a regular basis. However there are more dynamic influences on the road network, such as accidents, road works or big public events like parades. These can cause a lane or indeed a full link to be unavailable for traffic. Therefore a further attribute is needed describing whether a lane is blocked and if so why and for how long.

Connections

The approach taken so far implies that a vehicle can proceed from any incoming edge to any outgoing edge. As this is not always the case the connections within a junction need to be restricted. Figure 5 is a representation of a junction in Dublin city centre (Grafton Street, Nassau Street, St. Andrew Street) and depicts the dilemma: a junction with one incoming link from north, one outgoing to the west and two links allowing travel in both directions coming from east and south respectively. The arrows in Figure 5 (left) indicate all possible turning manoeuvres on this junction. However in reality this junction is heavily restricted: the southern link is a pedestrian area and therefore no traffic is allowed to turn in on this link. Furthermore due to the way the junction is constructed vehicles coming from the north are only allowed to turn left on the eastern link. Therefore Figure 5 (right) shows the correct (realistic) representation of this junction.

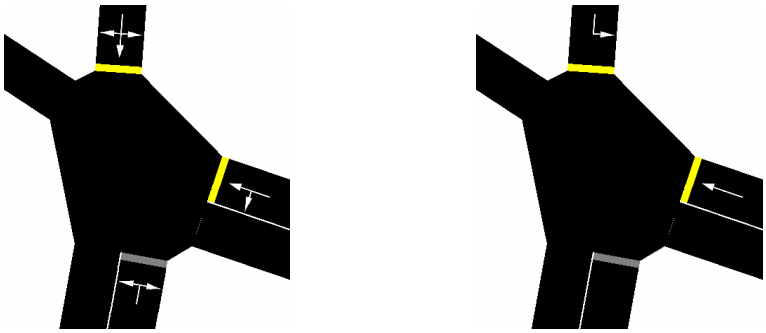

Figure 5, Legal turns on a junction in theory (left) and reality (right).

By now connections are only describing which incoming link is connected to which outgoing link. Additionally the connections can be used to describe which lane of an incoming link allows travelling to which lane of an outgoing link. This allows modelling lanes which are exclusive for vehicles turning in a specific direction. With this method exact replications of large, multi lane, multi link junctions can be build.

Bus Stops

As the main goal of this project is to determine the position of public transport vehicles in an urban environment one more feature of metropolitan road networks has to be incorporated into the model: stops where passengers can enter or leave the public transport vehicles. In our case we specify bus stops because the goal is to simulate buses in Dublin city but the model can also be applied to other means of transportation that share the road network with other traffic (e.g. trams).

Basically there two different kinds of bus stops: bus bays and lane based stops. A bus bay consists of an extra space adjacent to the link where the traffic is flowing. It has the advantage that the traffic flow is not interrupted while the bus waits for the passengers to enter or leave the bus. However they need more space than the lane based stops. Lane based stops on the other hand can be installed easily but hamper the traffic flow of the following vehicles. Both types of stops only have a limited capacity (i.e. the number of buses the stop can hold at one time). This can have an impact on the performance of the buses and on the traffic flow in general [34] (see section 3.2.3.2 for details).

3.2.1.2 Data Model

According to the description the data model as depicted in Figure 6 was created.

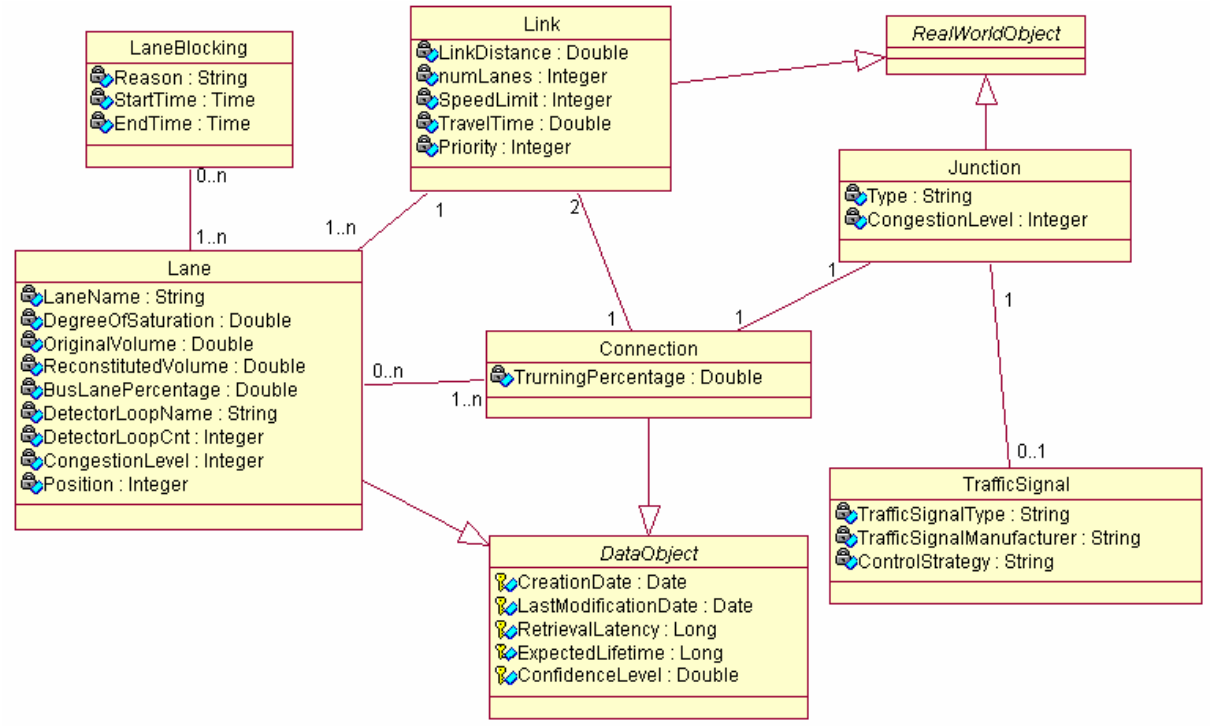

Figure 6, Road Network Data Model

3.2.2 Traffic

Now that the road network is described in sufficient detail to create a realistic model, this network needs to be populated with traffic. In this section a general model of traffic will be described which applies for both private and public transport. Features exclusive to public transport will be described in the next chapter.

As described in section 2.2.1 traffic can be modelled in an either macroscopic, mesoscopic or microscopic manner. We have chosen a microscopic modelling approach for this project because the main goal is to track single vehicles in the system.

There are several well studied microscopic car-following models, so we decided to use the model designed by Gipps and Kraus [24,32] as implemented in the SUMO traffic simulator. This model has the advantage that it can model traffic in urban environments, it covers both car-following and lane-changing behaviour. Furthermore it can deal with the imperfect behaviour of drivers and it is already implemented in the SUMO simulator [31,33].

3.2.2.1 Components

To model vehicle's according to the chosen model two aspects of a vehicle's trip need to be investigated: The vehicles behaviour on a link and at a junction. The former can be modelled using the physical properties of the vehicle and the car-following / lane-changing model. For the latter information on the vehicle's route through the network are needed.

Vehicles

The physical properties necessary to describe vehicles according to the Gipps / Kraus Model are as follows:

- Acceleration
- Deceleration
- Maximum speed
- Measure of driver's imperfection

Acceleration and deceleration are given in $m \cdot s^{-2}$ and describe the vehicle's ability to speed up and to slow down. The maximum speed in $m \cdot s^{-1}$ describes how fast a vehicle can travel, not with regards to the speed limits on a link but with regards to its physical capabilities. The driver's imperfection is the probability of a driver taking a specific action without a direct benefit from it, e.g. breaking for no reason or changing the lane although the one used before was not blocked.

In reality these properties vary for every vehicle. However for a microscopic simulation with several thousand vehicles it is hardly possible to gather all this data for each vehicle. To reduce complexity vehicles are grouped together as vehicle types where all vehicles of a certain type share the same physical properties. This allows averaging the properties for similar vehicles, but using different types allows modelling a wide range of vehicles. Examples for such types could be slow, medium or fast vehicles, defensive or aggressive drivers, cars, trucks or buses.

So far a model for traffic on a normal day, without interruptions in the service was presented. However external influences can have a big impact on a driver's behaviour. These external factors affect the properties described earlier. We decided to include weather as a factor in our model because it changes a driver's behaviour significantly. Bad weather conditions such as heavy rain, fog or snow and ice change mainly the physical properties of the car. Because

of the weather conditions drivers will accelerate more slowly and on a slippery or icy road it will take longer to stop the car because the tyres have less grip. Furthermore the average driver tends to be more cautious and to drive more defensive under unfavourable weather conditions.

Routes

By now the travel on a single link can be modelled. When the vehicle reaches a junction however, it needs to decide which way to go. In general a vehicle's journey is defined by a start point and an end point. These are links in the network. The intermediate links between start and destination form the route. The route which is chosen for a specific journey depends on several factors. Drivers mainly try to use the quickest route to get to their destination. Furthermore the choice of route is influenced by the personal preferences of the driver and his perception of the current traffic. For densely populated urban environments it is hardly possible to collect start and destination information from all the journeys that take place every day. Determining the routes for these journeys is even worse as they can change depending on the individual drivers of the vehicles. Therefore we decided to use a statistical approach to model the driving behaviour of private traffic (public transport usually uses fixed routes, for more details see chapter 3.2.3).

So to model vehicle trips statistically two measurements are important: for every link the number of vehicles entering or leaving the system on it and for every junction a probability measurement indicating the likelihood of a vehicle taking a specific turn.

The number of vehicles that enter or leave at a specific link could be generated using link inputs and outputs. Input and output describe the number of vehicles which enter and leave the link in a given time period through incoming and outgoing junctions respectively. Because we model an urban environment every link could be a source or destination for a journey [26] so the ratio of link input to link output is a good indicator of the likelihood of a vehicle starting or finishing its journey on this particular link.

Another possibility is to use the vehicle counts on a lane as provided by induction loops. The vehicle counts can be used to adapt the number of vehicles on each lane and therefore provide an equally precise instrument to recreate the traffic flow on a specific link. Compared to the input output ratio the loop count approach has the advantage that less data (one set per lane rather than two) is needed.

A junction turning ratio defines the likelihood for a vehicle on an incoming link to proceed to a specific outgoing link. E.g. on a typical crossing (possibility to drive on, turn left or right) the ratio could be: 10% left, 70% straight on, 20% right. Based on this data for each vehicle the decision which link to take next can be made.

From this data vehicles can be guided not by assuming fixed routes for fixed trips but by emitting and removing vehicles on links according to vehicle counts and then dynamically deciding how they proceed at each junction based on junction turning ratios. The loop count and the junction turning ratios are already part of the network model and therefore not presented in this model.

3.2.2.2 Data Model

According to the description the data model as depicted in Figure 7 was created.

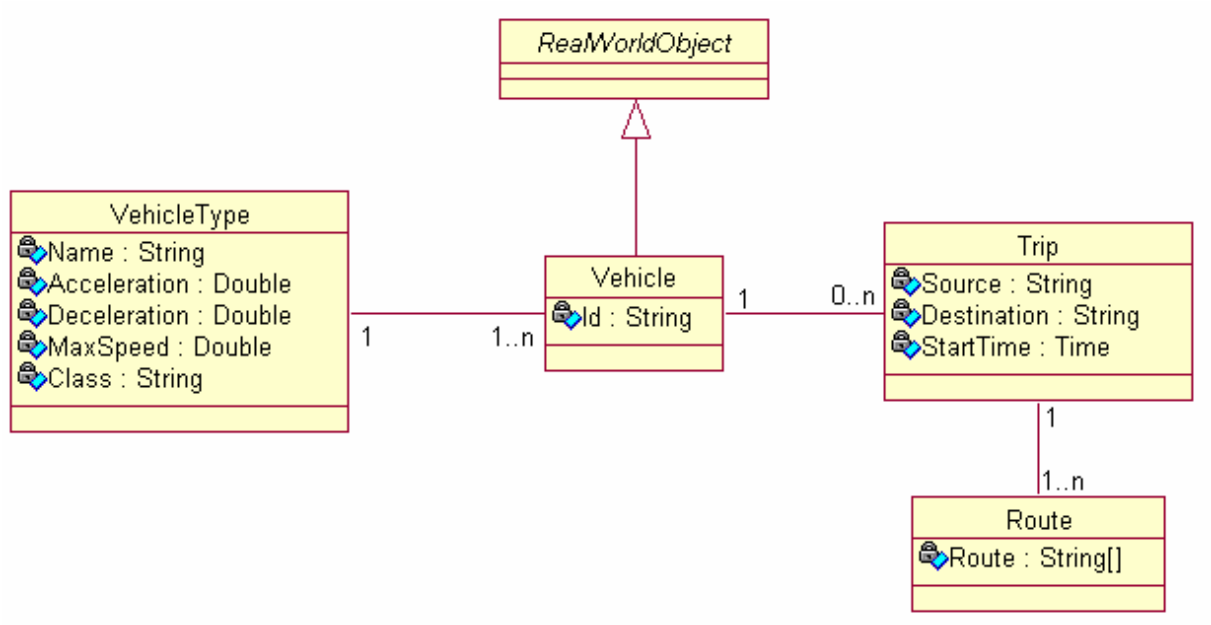

Figure 7, Vehicle Data Model

3.2.3 Public Transport

In the previous sections the models for the road network and for general traffic were explained. Public transport (PT) can be regarded as a specialised form of generic transport. In general PT vehicles adhere to the same car-following and lane-changing models as the private transport.

3.2.3.1 Components

The difference to generic traffic concerns the route generation and the stopping behaviour. In the following paragraphs we will detail a model describing a bus journey in the urban environment of Dublin city; however the model can also be used for other means of public transport in other urban areas.

Routes

In PT the routes are usually fixed. In the case of buses every bus of a specific bus route follows the same links from start to destination. The routes are predefined so that a particular set of bus stops can be served by the bus route. Temporary changes in the route only occur when some of its links are inaccessible while the service is running. Beside the routes the starting times are also fixed. Depending on the available timetables, information on when a bus serves a particular stop may be available. In Dublin this is not the case, only the time when a bus commences its journey is published in the timetables. Starting times and fixed routes eliminate the need for route generation based on statistics.

Stops

Apart from using fixed routes, stops are the second main difference to private transport. The behaviour of the PT vehicle at the stop does not differ from any other stop such as stopping for a red traffic light (i.e. slowing down, stopping, rejoining if traffic permits it). However these stops are not caused by the traffic conditions but by the need to get passengers into and out of the vehicle. The important characteristic of such a stop is that the waiting time depends on several factors. The main factors are the number of passengers that want to use a specific service, the ratio of people leaving at a specific stop and the frequency of vehicles serving a stop. The next section explains how this data can be used to model realistic stop times.

3.2.3.2 Stop Times

Fernandez and Tyler [34] have analysed the timing of buses at bus stops and divided it into the following sections:

- Queuing time
- Passenger service time
- Internal / external delay

Queuing Time

The queuing time describes the period a bus spends waiting when the bus stop is occupied. The queuing time is determined by the capacity of the bus stop and the number of preceding buses using the stop when the next bus arrives. To model queuing time the capacity of the bus stop and the number of preceding buses needs to be known. The capacity is explicitly modelled in the road network; the preceding buses are contained implicitly in the bus model as it represents all bus journeys in the city. Studying the behaviour of buses in Dublin we were able to find relatively short queuing times. The main reason is that most of the bus stops in Dublin are lane based, i.e. there are no extra bus bays at the stops. So stops have a relatively high capacity as buses just park behind each other, although this can lead to an extended dead time (see below). Furthermore the frequency of buses is rather low so that in general no more than three buses use a single stop at any given time. If there were more bus bays they could have a bigger impact on the queuing time as their capacity is usually limited to one or two buses. So any additional bus would have to wait for the bay to free up before it can let passengers enter or leave the bus.

Passenger Service Time

The passenger service time (PST) is the most important section of a bus stop as it describes the period when the bus is actually interacting with the passengers. Fernandez and Tyler [34] split up the PST into dead time and boarding time.

Dead Time

Dead time (t_{dead}) describes the time when passengers cannot interact with the bus, such as lowering the bus or opening / closing the doors. The dead time can be assumed to be fixed, in reality it might vary when the bus needs to unfold the wheelchair ramp or when the bus is parked behind another bus so that the passengers have to walk up to the bus before they can start boarding. However wheelchair access to buses is a rare occasion and can therefore be neglected in this model. Furthermore the effect of people forced to walk up to a bus is only a minor delay. While people walk up to the bus passengers who finished their journey can already leave the bus (serial boarding behaviour assumed, see below). Additionally those waiting at the stop usually see the bus arriving in advanced and start walking towards its stop (i.e. behind the bus currently waiting at the stop) even before their bus arrives. Thus this delay does not have a noticeable impact on is therefore not modelled within the dead time.

Boarding Time

The boarding time is the period when passenger are actually entering and leaving the bus. In Dublin a serial behaviour can be observed: First all passengers who have finished their journey leave the bus, then new passengers enter. The reason lies in the way Dublin's buses are designed: passengers have to enter and leave through the main door of the bus, i.e. the door beside the driver. Some buses have doors in the middle of the bus which would allow a simultaneous embarking and disembarking of the bus, however it is not used by drivers, only the main door is opened at the stops. Every passenger needs a different time to leave the bus or to enter the bus, pay his fare and clear the entrance. As it is not possible to get exact values of these times for every passenger the average time for embarking (t_E) and disembarking (t_D) is used in our model. This is the average time per passenger entering or leaving the bus. Thus to calculate the PST for a known number of passengers embarking (P_E) and disembarking (P_D) equation (1) can be used:

$$PST = t_{dead} + P_E \cdot t_E + P_D \cdot t_D \qquad (1)$$

The time periods t_{dead}, t_E and t_D can be determined empirically because they are average times and thus the same for all buses in the system. The number of passengers entering a specific bus depends on the total number of people using that particular bus route, the distribution of the arrival of passengers at the bus stop and the frequency of buses operating on this route. The number of passengers leaving the bus can be assumed proportional to the number of people on the bus.

Passenger Counts

The more people use a service the more will be waiting at the stop at any given time. The number of passengers arriving at the bus stop depends on the time of day and day of week, e.g. during peak hours in the morning and afternoon commuting people will mainly use public transport to get into or out of the city respectively. Apart from the number of passengers waiting at a bus stop it is also important to model when they arrive with respect to the bus they are trying to catch. With detailed timetables (i.e. departure times for each stop) most of the passengers tend to turn up at the stop close to arrival time whereas only few people arrive at the bus stop just after a bus was set to leave. This is effect is stronger with low bus frequencies because in this case arriving just after a bus left means a long waiting time [35]. However in Dublin departure information is not available for the stops on the way

but only for the start of a journey. Thus this behaviour is hardly found at Dublin's bus stops and can be neglected in this model. Instead the passengers are assumed to arrive uniformly at the stop over the course of time. The frequency of buses is important because the more buses arrive in a given time the less passengers are waiting for an individual bus. This aspect of the model is already covered by the timetables indicating the starting time of each bus.

The number of people leaving the bus can be described as a probability. Assuming there are no sudden changes in the behaviour of passengers the probability for any passenger to leave the bus at a specific bus stop can be given. Rather than using an absolute number this approach adapts to the number of passengers actually using the bus.

Internal / External Delay

Once all the passenger interaction is finished the bus has to rejoin the traffic. If it is hindered by another bus (i.e. the one parking right in front of it) the waiting time is referred to as internal delay. If the generic traffic is delaying the bus (e.g. when turning from a bus bay on a busy street) this is the external delay. Just as the queuing time this does not need to be modelled explicitly in the public transport model as the generic traffic model's car-following and lane-changing behaviour is capable of reproducing these delays.

3.2.3.3 Data Model

According to the description the data model as depicted in Figure 8 was created.

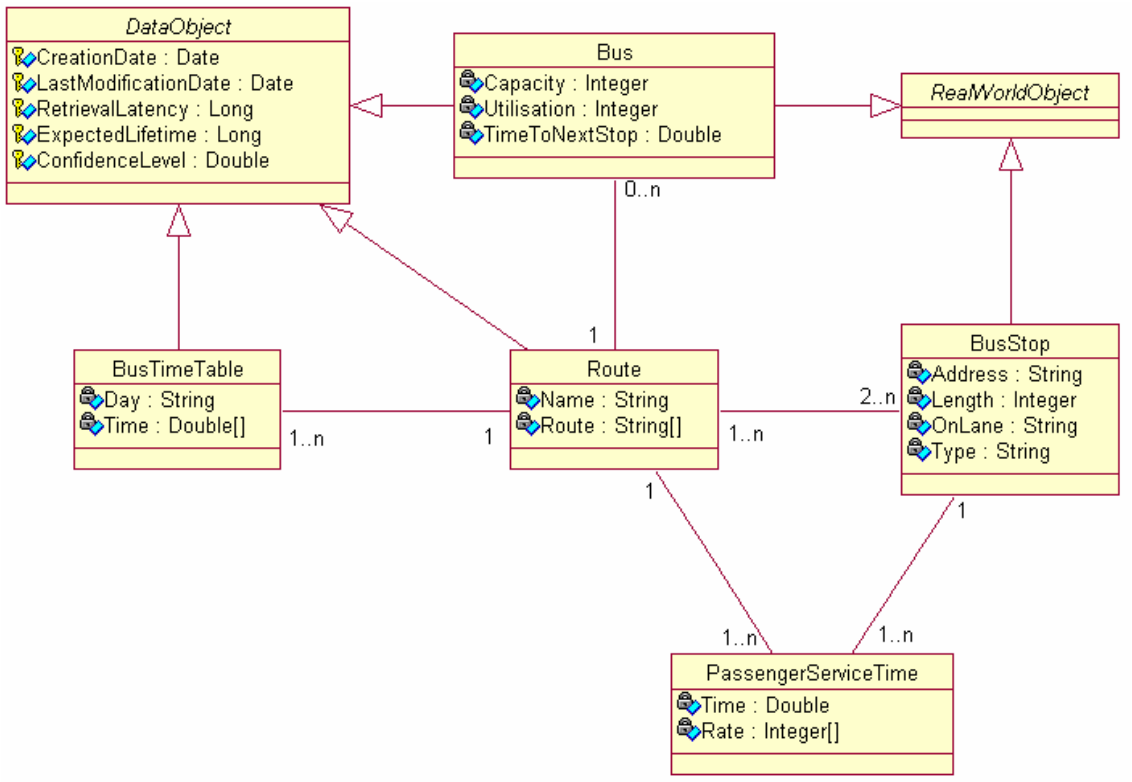

Figure 8, Bus Data Model

3.3 Online Simulation

To create an online representation of the current traffic situation it is necessary to adjust the number of vehicles in the simulation according to the induction loop counts provided by the ITS. Therefore the simulation has to be equipped with calibration points also referred to as calibrators. There are no calibrators in SUMO so we will describe a possible design in the next section.

Traffic Flow and Density

The traffic in a system can be described using flow and density [30]. The flow describes the number of vehicles that pass a point in a certain amount of time. Usually the flow is measured per minute or per hour. The density describes how many vehicles are present on a stretch of road of a certain length. The density can also be described as the reciprocal of the average gap between vehicles (i.e. if there are many vehicles on the road the gap between them is

rather small and vice versa). Flow and density can be combined in the fundamental diagram as shown in Figure 9.

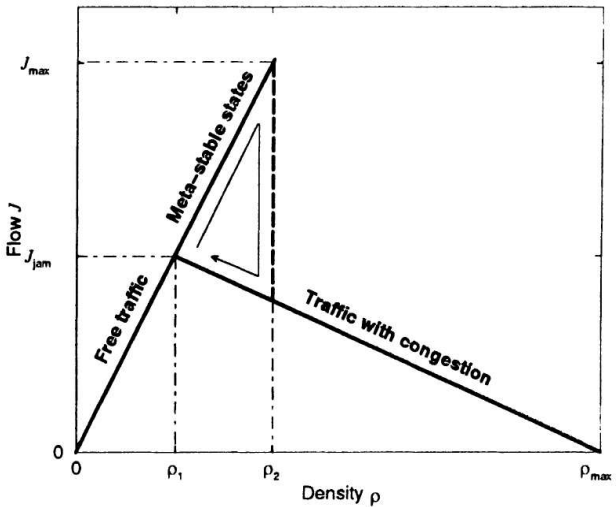

Figure 9, Traffic Flow - Fundamental Diagram [30]

The diagram shows three different states: free flowing traffic, congested traffic and a meta-stable state. The free flow starts off with a flow and a density of 0, i.e. no cars on the road. Once the density starts to rise, the flow increases as more and more vehicles are passing the road. However they are still sufficiently far apart from each other so they do not influence their successors driving behaviour. A further increase in the density lowers down the mean gap between vehicles further so that they start to influence each others driving behaviour. This behaviour is also represented in the car-following model widely applied in microscopic traffic simulation. Once the vehicles are sufficiently dense (density ρ_1 in the diagram) traffic jams start to emerge and the flow decreases. Now the level of congestion will rise with increasing density, down to the point where the maximum density (ρ_{max}) is reached. At this stage cars are standing 'bumper to bumper' and there is no movement and hence no flow at all. However Schreckenberg et al. [30] also describe a meta-stable state where the flow can actually rise over the critical value of J_{Jam}. In this state vehicles move in a synchronised fashion allowing for a high throughput [30 and references therein]. Once that synchrony gets disturbed the movement pattern changes to "stop-and-go", accompanied by a sharp drop in flow. This disturbance can be caused by the imperfect behaviour of a driver (e.g. overreaction and sharp breaking). The transition from a high flow meta-stable state to a lower flow congested state in irreversible. To increase the flow and overcome congestion the density of vehicles on this particular road must be reduced beyond ρ_1 to allow the traffic to flow freely again.

Calibration

Depending on the data available from the induction loops either flow or density of the traffic can be reconstructed. To calculate the flow simply the number of vehicles passing over the loop in a specific time period has to be counted. Schreckenberg et al. [30] have described several ways to estimate the density of traffic; however for this approach further data is needed such as the time each vehicle spends on the induction loop or the velocity of each vehicle.

In this project the calibration will be based on flow estimation. We decided to use this approach for several reasons: Firstly less data needs to be retrieved. Secondly an adaptation of flows in the simulator is much easier to achieve than a density calibration. Lastly online simulations of the city of Duisburg have shown that such an approach is sufficient to create a realistic online simulation of an urban environment [30]. So to realise a flow estimation the actual vehicle counts and the length of measuring periods have to be retrieved from the iTransIT system.

4 Implementation

To prove the feasibility of the design described an implementation is necessary. Therefore the following chapter describes the general implementation of the iTranSIM system, the changes necessary to the simulator SUMO to allow online simulations and details on the proof-of-concept realization of this implementation.

The system design as described in chapter 3 was slightly modified to accommodate the input formats accepted by SUMO. Mainly the "Data Update" component of iTranSIM-DP was split into a component dealing with static input data and one dealing with dynamic input data. The modified design is depicted in Figure 10.

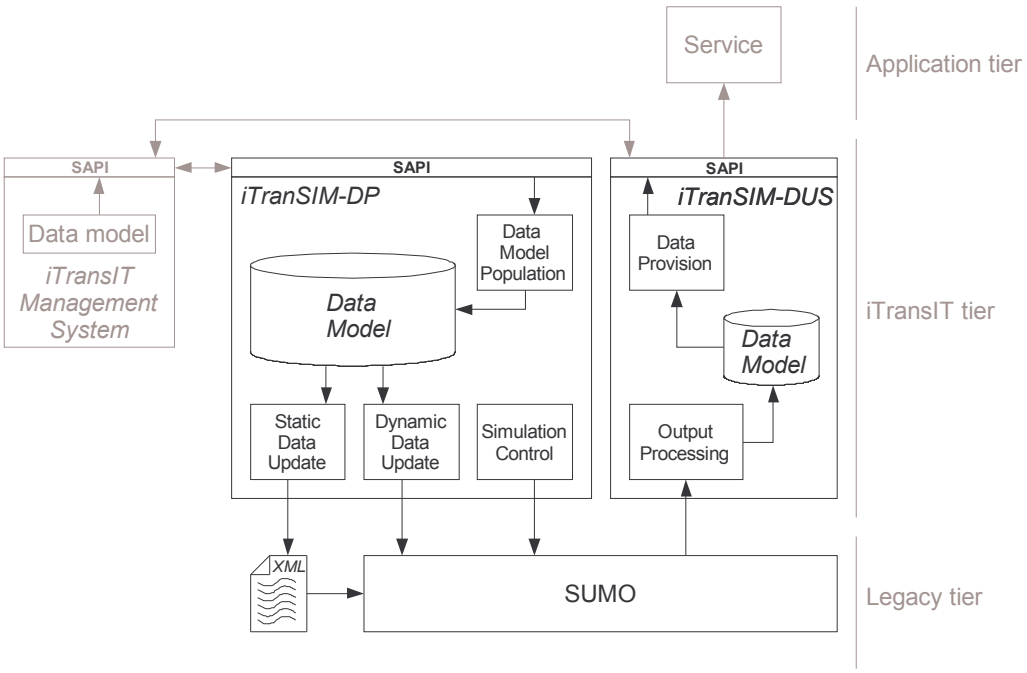

Figure 10, iTranSIM System Implementation

The two iTranSIM systems now consist of 6 main components:

- Data Model Population: The data relevant to the simulation is fetched using the spatial API.
- Static Data Update: All the static data which is necessary for the simulation is retrieved from the data model and converted into XML files which SUMO uses as input.
- Dynamic Data Update: Dynamic Data such as the current detector loop counts are acquired and fed into SUMO on a regular basis to ensure an up-to-date representation of the current traffic situation.

- Simulation Control: This component is used to assure that the simulation is running correctly and synchronised to the incoming traffic data.
- Output Processing: The relevant data (i.e. the positions of buses) is received form SUMO and stored in the data model.
- Data Provision: The location data is provided to other services using the spatial API.

In the following sections these components will be discussed.

4.1 iTranSIM-DP

4.1.1 Data Model Population

To populate the data model the spatial API is used. The data is fetched from the iTransIT management system and stored in two container objects. The static data container is filled once and provides data for the static data update component. The dynamic data container is updated regularly and provides data to the dynamic data update. Both classes are depicted in Figure 11.

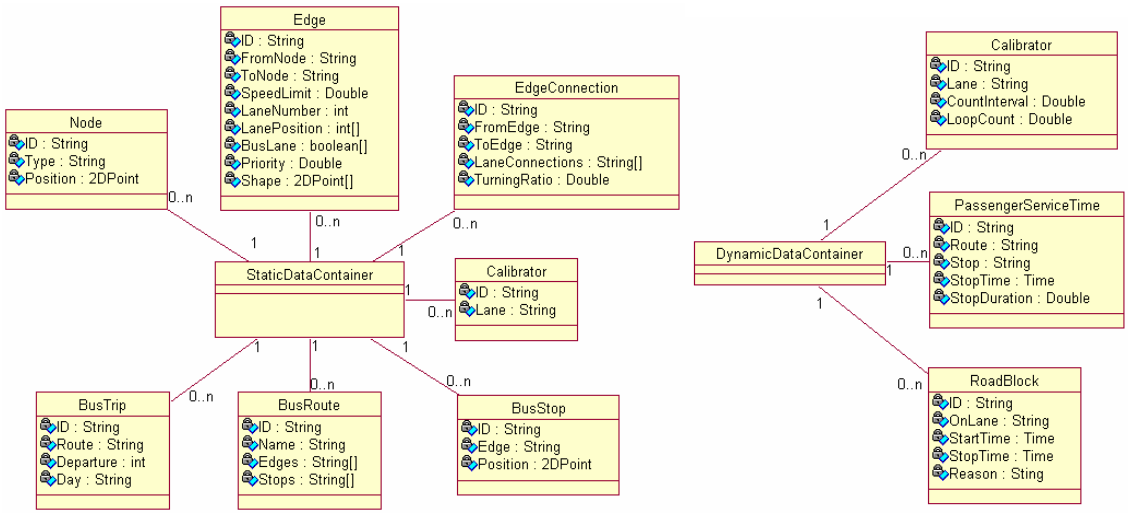

Figure 11, iTranSIM - Data Container

4.1.2 Static Data Update

To run a simulation in SUMO a minimum of input files is needed. All SUMO input files are in XML format, so they are both human and machine readable. The following sections describe which files are needed for a simulation and how the information can be obtained and delivered to SUMO.

4.1.2.1 Road Network

To setup the road network a network file is needed. This file contains most of the information described in section 3.2.1 such as junctions, links, lanes, connections and traffic lights. As this is a very complex dataset network files are not meant to be build by the user. Instead SUMO provides a tool called netconvert which generates the network file from a number of input files. The most important input files are:

- Node file: This file contains information on junctions, i.e. their position in the grid and their type.
- Edge file: In this file all the data on links and lanes is stored: position (either as coordinates or as a pair of nodes), number of lanes, maximum speed, priority for unregulated junctions, length and a set of coordinates to describe their shape. Furthermore the vehicle types which are allowed or disallowed on specific lanes are stored in this file.
- Connection file: In here specific information on allowed turning manoeuvres is stored, i.e. which link connects to which link or on a more detailed level which lane connects to which lane.

All this information constitutes the bare road network. The data can be fetched from the data model and then transformed into the necessary XML format. Then netconvert is called with these files as input to generate the network file.

Other information regarding the road network is not part of the network file itself but it is stored in so called additional files. Additional files contain information about:

- Traffic light programs: A representation of the light changing cycles on particular traffic lights.
- Variable speed signs: These allow adapting the maximum speed on a lane. Although this setup was neglected in the model it still proves useful for the simulation: When setting the maximum velocity to 0 this blocks the lane because vehicles in the model strictly adhere to the speed limits.
- Detectors: As described in 2.2.4.2 there are several types of detectors in SUMO. These are necessary to generate output such as loop counts which are necessary to evaluate the simulation.
- Bus stops: The position and length of bus stops is also defined in additional files.

Again these information can be obtained from the data model. The generated files are then used as an input for the SUMO simulator.

4.1.2.2 Static Traffic Information

With regard to the traffic there are two kinds of data sets: those who are supposed to be fixed such as the timetable of the bus, the different vehicle types in the system. On the other hand there is dynamic information that needs to be adapted while the simulation is running such as the number of vehicles in the network or the state of road blocks which are caused by accidents or road works. In this section the processing of static data is explained, the dynamic aspect of the traffic is detailed in the next section.

Dynamic Vehicle Guidance vs. Vehicle Routes

As mentioned earlier in an urban environment it is not possible to know all the trips people take and the routes they choose to get to their destination. Junction turning ratios are a good indicator for a driver's behaviour on a junction if no explicit routes are given. However SUMO does not allow dynamic vehicle guidance based on junction turning ratios. Every vehicle in the simulation needs to start out with a fixed route.

Although there are re-router elements that allow changing the route of a vehicle while the simulation is active, they are not ideal for guiding vehicles dynamically through the road network either. Re-routers either have to be supplied with a new route for a vehicle, or a new target link where the route to that link would be calculated using Dijkstra's shortest path algorithm. However it would not suffice to supply the next edge as the target because SUMO would then remove the vehicle once it leaves the current link, assuming the final destination is reached. Thus to use re-routers for truly dynamic vehicle guidance based on junction turning ratios for each junction a set of routes has to be calculated and a re-router has to be placed on every edge.

Alternatively a set of static routes can be generated for all possible starting points based on junction turning ratios. Starting points in this case are all the calibrators, i.e. the links where vehicle counts get adapted to real world data. The rationale behind this approach is that although vehicles could be guided dynamically using re-routers this would cause a big overhead.

Generating Vehicle Routes

A tool called jtrrouter which comes with the SUMO package is used to create a set of routes that would be common for a starting point based on the junction turning ratios provided. These routes will then be used by the vehicles added into the system based on induction loop counts (see below). The following input files are needed for the jtrrouter:

- Network file: The file describing the network as detailed above.
- Flow file: The number of vehicles as well as their type and where they start (link).
- Turn file: The turning ratios for the connections in the network.

The result is a route file which could be used as an input into SUMO. However the route file not only specifies possible routes for specific starting points but also a plan on when to emit vehicles on these routes. The starting times are set according to the number of vehicles that were described in the flow file (e.g. if a flow of 120 vehicles per hour was specified the plan contains a vehicle emission every 30 seconds). Thus this file can be used in SUMO to generate a predefined amount of traffic in a system. However the aim of this project is to create an online simulation, i.e. to generate the amount of vehicles dynamically. Thus the route file generated by the jtrrouter needs to be processed further. The route descriptions have to be extracted and stored in a separate file so they can be assigned to the vehicles emitted into the system dynamically.

Additional Data

Apart from the routes for the private vehicles other information that needs to be converted into SUMO input files include:

- Vehicle types: On one hand these types are used to model different vehicle properties such as acceleration and deceleration. On the other hand lanes can be allowed or blocked for certain vehicle types.
- Bus stops: The position and length of bus stops.
- Bus routes: The set of links a bus travels on during a journey and the bus stops served.
- Bus trips: Starting times for buses.

As with the data on the road network all of the relevant information can be obtained from the data model. For the starting points in the flow file (jtrrouter input) simply the links are chosen where later the traffic has to be adapted to the real induction loop counts. From that input the

jtrrouter generates a set of routes that can later be used when vehicles are emitted at a calibrator.

4.1.3 Dynamic Data Update

To generate a realistic, online Simulation of Dublin the number of vehicles in the system needs to be updated regularly. The following data is relevant for the simulation and dynamically changing over time:

- Induction loop counts: The most important information for generating a realistic online simulation. These values are used to adapt the flow of vehicles in the simulation so that the number of vehicles in the simulator actually matches the amount of vehicles in the real world.
- Blocked roads: If certain lanes or links are blocked they need to be cordoned off in the simulation as well. However this only applies to situation where external influences cause the road to be blocked, i.e. it does apply if there are road works going on or a vehicle broke down but not for normal traffic jams as they are modelled implicitly by the simulator using the real vehicle counts.
- Passenger service times: To create a more realistic reproduction of bus journeys in the simulator the times needed to serve passengers at bus stops can be used as a dynamic input too. This allows modelling of situations where two buses arrive very close to one another or where the drop out of one bus affects the service time of the following bus as it has to deal with more passengers at the stop.

The data is provided by the data model and regularly updated by the data model population component.

Dynamic Data Input into SUMO

To input this information into SUMO a new interface needs to be created as SUMO currently only allows for static input via files. This is useful for statistical analysis or evaluation of traffic management strategies but it does not allow online simulations. To update the dynamic data periodically some kind of inter-process-communication (IPC) needs to be employed. The two systems involved are based on two different technologies: SUMO is based on C++, iTranSIM is written in Java. Thus IPC solutions which are proprietary to a single technology such as COM or JavaRMI can not be used. Furthermore the solution must support distributed data exchange because iTranSIM and SUMO might not be running on the same machine. On

the other hand the set of data to be transferred from iTranSIM to SUMO is well defined and rather small. Additionally the communication takes place between two processes only, both of which we have control over. Therefore common features of distributed technologies such as concurrency control or transaction management are an unnecessary overhead and thus neglected in this approach.

As a result of the observations stated above we decided to use a socket interface as it provides a good balance between simplicity, functionality and the ability to distribute the processes involved in the communication. To update the data SUMO will act as a server, listening on a socket and storing the information internally. From there the relevant classes within the SUMO application can use the data to update the simulation. The iTranSIM on the other hand will periodically poll the spatial API (as it is a synchronous API). Once the data is obtained it will be processed and transformed into a format which can be handled by SUMO. Finally a TCP connection will be established to the SUMO socket server and the data is submitted.

4.1.4 Simulation Control

This component is necessary to keep the simulation synchronised to the real time and to deal with possible errors in SUMO.

The Simulator itself will be restarted every 24 hours, this way changes in static data (e.g. bus routes or timetables) will be accommodated on time. The restart should take place at a time when the operation of the system is not affected. For this project the restart will take place at 5 am, because at this time no buses are on duty in the city (the last nitelink night bus leaves city centre at 4 am and regular service does not commence before 5:30 am).

To get accurate results it is vital to keep the simulation in sync with the real world. With the simulator's clock running slightly faster or slower than one second per simulation step the difference would accumulate over the day and results would be inaccurate.

SUMO does not offer precise timing features as it is meant to perform offline simulations from a static input. For those simulations it is important to run as fast as possible so the results can be analysed further. For our project however the execution speed is not important as long as the simulator is capable of calculating one simulation step in less than a one second.

Remote Controlling SUMO

We decided not to implement the timing features in SUMO but rather to use a component called itm remoteserver (named after the developers, the Institut für Telematik at the University of Lübeck, Germany) [36]. The itm remoteserver allows controlling the behaviour of SUMO using a socket interface.

The function relevant to our project is called "simulate step". The server interrupts the execution of the simulation and waits for an incoming packet. In that packet the number of simulation steps to be executed is specified. Once the server received such a packet it will start the simulation, pause again when the number of steps specified in the packet is reached and wait for the next packet.

So instead of implementing a timing component in SUMO we implement it in the iTranSIM. From here the simulator is controlled using the itm remote server. An extra thread will trigger the execution of the next steps periodically. This gives us more control over the simulation and more flexibility than a timing component in SUMO could.

Every single time step can be triggered individually, thus a one second timer in the iTranSIM system is used. Alternatively a block of simulation steps can be triggered at once, e.g. 10 steps at a time, triggered every 10 seconds. This makes sense during periods when no traffic flow updates arrive from the detector loops. For example if the traffic flows are adapted to real traffic data every 20 seconds, it makes no difference whether the 20 steps after an update are triggered once every second or en bloc. This gives the opportunity to distribute processing time and CPU usage for the different processes which could become necessary if there is a large urban environment to be simulated, but it complicates the output processing as it needs to map the results to a time when they actually become valid.

4.2 iTranSIM-DUS

4.2.1 Output Processing

Section 2.2.4.2 described what output data can be generated by SUMO. Usually this data is written to XML files so it can be analysed after the simulation has finished. However there is also the possibility to specify a TCP port rather than a filename to direct the output to.

This feature can be utilised for the processing of the output. The initial idea was to use the network state dump to locate the buses in the system. In that dump the states off all lanes, including the vehicles on them, are provided. Thus the network state dump needs to be parsed

to find the relevant information on buses. From the lane on which the bus is driving and the position in the lane (measured in meters from the start of the lane) the geographical coordinates of the bus can be calculated. Furthermore the speed of the bus is provided and can be used to indicate whether the bus is driving or standing somewhere.

As the network state dump gets very large, particularly when dealing with detailed simulations and large quantities of cars, this approach does not scale very well. After some testing on this initial implementation we found this approach unfeasible because the time needed to process the network state dump even for a medium sized simulation exceeded the 1 second that a simulation step is allowed to take at the maximum. Therefore a vehicle type probe was implemented in SUMO which provides us exactly with the information we need, i.e. the status of buses and their positions. The probe is described in detail in section 4.3.2.

The position and the speed of the bus are then updated in the data model. The position of a bus can be updated every second (i.e. after every time step). However if SUMO is simulating time blocks (see above) the data received from the simulator represents a value from the near future. Thus the positions need to be mapped to a timestamp so they can be updated right on time. We use the time step data which is stored in the vehicle type probe dump as a mapping criterion to map simulation time to real time and to schedule the update of the bus positions in the data model.

4.2.2 Data Provision

This component is responsible to provide the data acquired from the simulation to the services in the application tier or to other iTransIT system. To comply with the iTransIT framework the data provision has to implement the functionality of the spatial API.

When a service or an iTransIT system needs access to the data it can use the same spatial API calls that are also used to access the data e.g. from the iTransIT management system. The data provision component then obtains the data from the internal data model and forwards it to the system that requested it.

Alternatively the data could also be stored in the iTransIT management system by updating the spatial database using the `S_API::update()` method. This way the services or iTransIT systems would have to request the data from the iTransIT management system rather than from the iTranSIM-DUS. This would keep the complexity of the iTranSIM-DUS low because no implementation of the spatial API is necessary. However the load on the

iTransIT management system would be increased and the data model in the management system would have to be adapted to store the location information of buses.

4.3 SUMO

4.3.1 Calibrators in SUMO

To adapt the vehicle counts to real world data a calibrator needs to be implemented in SUMO. The calibrator used for this project will adapt the vehicle flow (as opposed to the density) according to detector loops counts provided.

The calibrator added into SUMO is loosely based on the SUMO-emitter. Just as the emitter the calibrator is a trigger element. A trigger executes a specific function with a given frequency; the emitter for example emits vehicles into the system based on its configuration. Other triggers are re-routers or variable speed signs which either changes a vehicles route or the maximal speed on a link for every vehicle that reaches the link.

Configuration

The calibrator is configured the same way all other triggers are configured, using XML files.

```
<trigger id="calibrator-1309-1305_0" objecttype="calibrator" pos="1"
         objectid="1309-1305_0"
         file="X:\iTranSIM\test_1309-1305.cal.def.xml" />
```

Listing 1, Calibrator Configuration

Listing 1 shows the configuration for a calibrator:

- id: a unique identifier
- objecttype: the type of trigger, a calibrator in this case
- pos: position on the lane, 1 m after the start of the lane in this example
- objectid: identifier of the lane on which the calibrator is located
- file: file containing possible routes for newly emitted vehicles, an example is listed in Listing 2

```
<calibrator>
   <vtypedistelem id="CAR" probability="1"/>

   <routedistelem id="vRoute-1309-1305-1" probability="0.4959"/>
   <routedistelem id="vRoute-1309-1305-2" probability="0.0801"/>
   <!-- more routes here -->

</calibrator>
```

Listing 2, Calibrator Definition

The vehicle type distribution (vtypedistelem) describes the probability for an emitted vehicle to have a certain vehicle type. The route distribution (routedistelem) provides probabilities for a vehicle to use a specific route. The routes are generated using junction turning ratios and the jtrrouter tool provided by SUMO (See 4.4.2).

Function

The main functions of the calibrator are removing vehicles when there are too many in the simulator or emitting additional vehicles if there are too few in the simulation. Thus the calibrator actually has to "know" how many vehicles passed over it. This value is then compared to the counts delivered by real world induction loops. To count the vehicles in the simulation an induction loop element is create at the position of the calibrator. This element is similar to the E1-detector (virtual induction loop) only that count values can be obtained internally and are not written to a file.

A function is then registered to be executed every second; this function implements the main logic of comparing real to virtual vehicle counts and emitting or removing vehicles. The calibrator receives loop counts and the interval over which these counts are taken (e.g. 5 vehicles over the last 60 seconds) as input. This value is broken down into the amount of vehicles per second, which is usually a value smaller 1. Thus every second this value gets added up until it is greater than one, i.e. one vehicle is supposed to pass the calibrator. When this real word value is greater than the count from the calibrator a new vehicle is scheduled to be emitted. If it is smaller then the last vehicle that passed the calibrator is removed. Because the induction loop in SUMO does not deliver vehicle IDs of vehicles passed, to remove a vehicle the list of all vehicles on a lane is analysed, finding the first one that is located behind the calibrator and eventually removing it. If the vehicle has already left the lane (i.e. there are no vehicles left behind the calibrator), an extra vehicle will be removed in the next simulation step. Therefore calibrators should always be located at the start of a lane so vehicles are unlikely to pass the calibrator and leave the lane in one step. The minimum distance to the

end of the lane should be 14 m so that at a maximum speed of 50 $km \cdot h^{-1}$ or 13,9 $m \cdot s^{-1}$ the vehicle is still on the link.

4.3.2 Vehicle Type Probe

As mentioned in 4.2.1 the initial approach to retrieve vehicle positions from the network dump was inefficient and not feasible even for smaller simulations. Therefore a vehicle type probe was implemented to generate exactly the output needed.

Configuration

As with the calibrator the vehicle type probe trigger is configured using XML files, an example is given in Listing 3.

```
<trigger id="bus_probe_socket" objecttype="vtype_probe"
         type="BUS" freq="10" file="localhost:1339" />
```

Listing 3, Vehicle Type Probe Configuration

The elements of the vehicle type probe are as following:

- id: a unique identifier
- objecttype: the type of trigger, a vehicle type probe in this case
- type: the type of vehicle that should be monitored
- freq: the frequency at which the output is generated (every 10 seconds in this case)
- file: the file which the output is written to, in this case it is written to a socket. An example output is shown in Listing 4

```
<vehicle-type-probes type="BUS">
   <timestep time="10">
      <vehicle id="b-15-i-1" edge="5434-689" lane="5434-689_0"
               pos_on_lane="36.54" x="513.83" y="36.26"
               speed="4.74"/>
      <!-- more vehicles here -->
   </timestep>
   <!-- more timesteps here -->
</vehicle-type-probes>
```

Listing 4, Vehicle Type Probe Output

Implementation

In SUMO all vehicles are managed by the `MSVehicleControl` object. The vehicle type probe queries this object for all active vehicles (i.e. all vehicles that are currently on a trip in the simulation) and checks for the vehicle type of each of those vehicles. If the type matches the type required to monitor, the properties of the vehicle in question are read and the output is written to the file or socket.

This implementation allows gathering the information required much faster then parsing the network dump. It is also sufficiently fast to allow online simulation within our tests.

4.3.3 Remote Server

The remote server used to control the simulation and update the data is an enhancement of the itm remote server implemented by the institute of telematics in Lübeck Germany. There are two commands used by iTranSIM-DP: 'Simulate Step' and 'Update Calibrator'.

The server listens on a TCP socket fir incoming connections. The messages are in a binary format which will be described in this section.

Message

A message can contain several commands; the structure of a message is depicted in Figure 12.

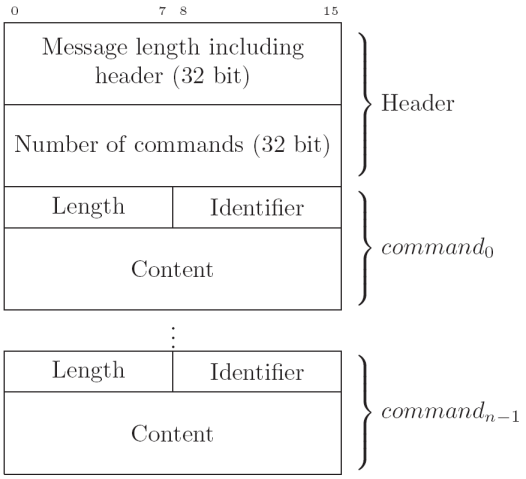

Figure 12, Remote Server Message [36]

The advantage of sending several commands in one block is that due to the slow start algorithm in TCP the transfer becomes faster for longer messages. Furthermore there is only one TCP header. This header with a length of at least 20 bytes creates a relatively big

overhead compared to the message length of the remote server messages ('Simulate Step' is 15 bytes long). Thus sending one message with 10 commands in it is faster than sending 10 messages with one command each.

Simulate Step

The 'Simulate Step' command allows to run a simulation up to the specified target time. Its structure is depicted in Figure 13.

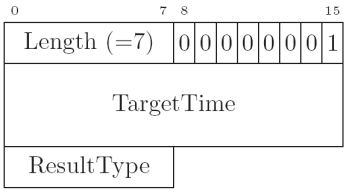

Figure 13, Remote Server - Simulate Step [36]

The `TargetTime` is a 32 bit integer describing the timestep to which SUMO is supposed to run. The `ResultType` describes what data should be included into the response that the server sends to the client. This stems from the original purpose of the itm remoteserver, for this project we have added a result type 0xFF which produces no output.

Once the remote server receives such a packet it calls the `SimulateStep()` method until the target time is reached.

Update Calibrator

This command is used to update the data for a calibrator. The message contains the vehicle count, the time period over which the measurement was taken and the ID of the calibrator. Because the ID can be a string of arbitrary length this length also has to be transmitted. The structure is depicted in Figure 14.

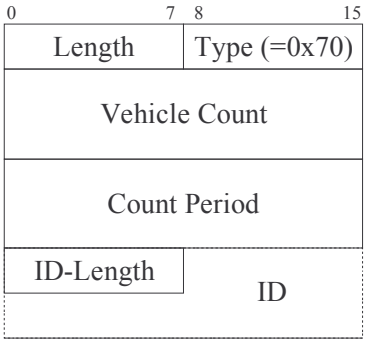

Figure 14, Remote Server - Update Calibrator

On receiving of an 'Update Calibrator' message the data is extracted from the message and the static method `MSCalibrator::updateCalibrator(String id, int duration, float count)` is called to update the values. In this method the calibrator with the given ID is updated, an error code is returned if the Calibrator cannot be found.

4.4 Proof-of-Concept Implementation

According to the design and the implementation described above a proof of concept was implemented to provide data for evaluating the design. In the following section the general realisation of the prototype is described alongside with specific problems we encountered during the implementation.

4.4.1 Information Retrieval Using the Spatial API

The iTranSIM system stores all the necessary data in objects according to its data model. Most of the data is retrieved from existing iTransIT systems using the spatial API. The objects are stored in one container object which uses a populator object to retrieve the data and to store it in the container. In addition to an SAPI populator there is also a JDBC populator implemented. This allows retrieving the data directly from the spatial database and was used mainly for testing and evaluation purposes. According to the specification in section 4.1.1 two data containers were created, one for static and one for dynamic data.

In the following section the population of the data container with junction objects using the spatial API will be explained exemplarily for all populators.

Retrieving Junction Data

To retrieve the information about junctions first the objectIDs for all junctions have to be retrieved. This is done using a call to `S_API::select(String elementType)`, requesting all objectIDs for the element type "R_D_Junction". Then for each objectID `S_API::retrieve(String elementType, int objectID, String[] parName)` is called to retrieve the parameters "ID", "juncType" and "Geometry" from the spatial database. The ID is also used as an ID in the SUMO node file, the juncType defines whether a junction is controlled by a traffic light or priority based. The geometry contains the x and y values describing the position of the junction in the map. As the data formats describing geometry in iTransIT and SUMO are compatible no extra transformation was necessary. See paragraph Location Mapping (below) for details.

For each junction the data is stored in a Node object as depicted in Figure 11, page 53. The data container holds a vector of Node objects to store the data about all nodes. Once the data is retrieved and stored for all junctions the other objects are retrieved and stored in the data container. After all the objects are stored the data is transformed into SUMO XML input files as described in 4.1.2.

Location Mapping

One of the concerns when starting the project was how we could transform the coordinates of junctions, links and vehicles from the format they have in iTransIT into a format understood by SUMO.

SUMO uses a two dimensional, Cartesian coordinate system as its basis, with the origin in the lower left corner. The values of x and y are given in meters from that origin. There is also the possibility to provide SUMO with geo-coordinates, i.e. longitude and latitude in degrees.

The geo data in iTransIT on the other hand is encoded using the Ordnance Survey Ireland mapping datum (OSI mapping). This system is specific to Ireland and splits up the island's map into sub-grids (as depicted in Figure 15). Just as SUMO it uses a Cartesian coordinate system, with the origin in lower left (south-western) corner of a sub-grid and the x and y values are also measured in meters.

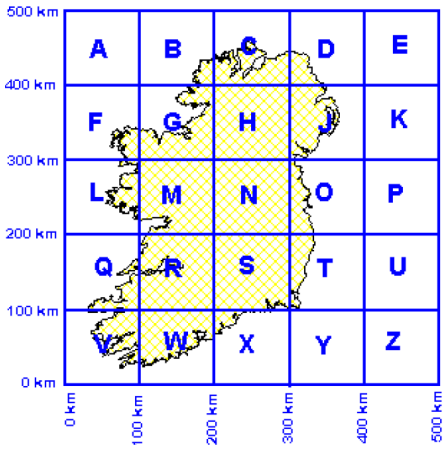

Figure 15, OSI Grid System Ireland [6]

As both systems iTransIT and SUMO use the same principal representation for their coordinates a mapping from one system to another is not necessary. The data retrieved from the iTransIT system (x, y coordinates according to the OSI mapping) can be used directly in the sumo node and edge files (x, y coordinates of junctions, waypoints for links)and for representing the vehicle positions.

4.4.2 Route Generation

As described in section 2.2.4.2 SUMO does not support dynamic vehicle guidance based on junction turning ratios. Therefore an adequate set of routes needs to be generated for each link that is equipped with a calibrator. These routes are not stored in any iTransIT system, therefore they can not be retrieved using the spatial API. However the jtrrouter tool allows building such routes, but its output is a flow definition file, containing a route based on turning ratios for every vehicle that is to be emitted. An excerpt from such a route file is shown in Listing 5.

```
<vehicle id="1_8_0" type="CAR" depart="101">
   <route>1309-1305 1305-678 678-1314 1314-1313</route>
</vehicle>
<vehicle id="1_7_0" type="CAR" depart="101">
   <route>1309-1305 1305-678 678-679 679-9400 9400-1298</route>
</vehicle>
<!-- more routes here -->
```

Listing 5, SUMO Route File

Using the jtrrouter a flow file was generated with 10000 vehicles per starting point, starting points being the lanes where calibrators are located. From this file the route information was extracted, discarding the vehicle information. The routes were then sorted and their probability was calculated by counting the occurrence of the route, i.e. if 100 vehicles out of the 10000 use the same route, this route has a probability of 0.01 or 1%. The extraction, sorting and counting was implemented using a very simple shell script and the UNIX tools sed, sort and uniq. The script takes the file produced by jtrrouter and writes its results (routes and probabilities) into a file supplied as the second argument, using a CSV style format. The script is shown in listing Listing 6.

```
#!/bin/sh

sed -n -e "s/^.*<route>\(.*\)<\/route>.*$/\1/p" $1 | \\
      sort | uniq -c | sed -e "s/\\t/ /g" -e "s/^ *//" > $2
```

Listing 6, Shell Script for Route Extraction

The output file is then read by the iTranSIM system and used to generate the route file (containing the routes) and the calibrator definition files (containing the probabilities for these routes). An example route file is shown in Listing 7, the calibrator definition file is shown in Listing 2, page 61.

```
<routes>
   <route id="vRoute-1236-1237-1" multi_ref="x">
        1236-1237 1237-1300 1300-1299 1299-675 675-1303 1303-1304
   </route>
   <!-- more routes here -->
</routes>
```

Listing 7, Calibrator Route File

4.4.3 Simulation Control

It is important to ensure the timely execution of all simulation steps, as described in section 4.3.3 we use the itm remoteserver to send the instruction "simulate step" to SUMO every second. Alongside this command the updates for the calibrators are sent to SUMO, so the values for the vehicle counts are up to date.

For the correct timing a simple mechanism was implemented to send the command to SUMO every second. Although there is a timer in Java we decided to implement our own timing code because the timer available in Java would start a new thread when executing the function associated with the timer. Therefore when it comes to delays in the system, e.g. due to garbage collection, then several simulator updates might occur simultaneously and be executed in the wrong order. Therefore our solution is strictly serial, if it comes to delays in the execution of the code the updates will be sent to the simulator back to back until timeliness is back to normal, but the updates are guaranteed to stay in the right order. The code of the timing function is presented in Listing 8.

```
int i2 = 0;

long startTime = System.currentTimeMillis();

for(int i = start; i <= stop; i++) {
   i2++;

   myDynamicDataContainer.update();
   sendCalibUpdate();
   sendSimStep(i);

   long time2sleep = (i2 * 1000 + startTime) -
                     System.currentTimeMillis();

   if(time2sleep > 0){
      Thread.sleep(time2sleep);
   }
}
```

Listing 8, SUMO Control Timing

Before the loop starts to send the message to SUMO the current time is taken and stored in `startTime`. Then for every step the time when it is supposed to be finished is calculated based on the `startTime` and the number of steps previously executed. The difference between the time when the step is supposed to be finished and the current time is then used to put the thread into sleep mode. If there is no difference, or current time is already greater then the supposed stop time, then the thread does not sleep and continues straight away, this allows for the catching up with the real time if the system got delayed for some reason.

4.4.4 Output Processing

The output generated by the vehicle type probe has to be parsed and stored so other services can access it through the spatial API. For this proof-of-concept the data was not stored in the iTranSIM-DUS system but it was also stored in the spatial database. The data can still be queried by services, however these services have to use inquire the iTransIT management system rather than the itranSIM-DUS. Similar to the populators used to retrieve data using the spatial API, this service implements an updater objects that allows updating the vehicle positions into the spatial database using the spatial API. Again there is also a JDBC updater for direct access to the database which was mainly used for testing.

To parse the output a SAX parser was created with a customized SAX handler to react on the elements that can occur in the vehicle type probe output, namely "timestep" and "vehicle". If a new timestep element is detected its element data is stored. Then for every vehicle element found in this timestep, the data is updated using either the spatial API or the JDBC updater.

Alternatively to the SAX parser for the XML a processor for binary input was also discussed. If the vehicle type probe would provide a binary format rather than the XML format the amount of communicated data and necessary processing could be reduced. However this was not implemented in this initial proof-of-concept because the SAX parser was able to handle the XML output with a sufficient performance. For a detailed evaluation see section 5.4.3 in the evaluation chapter.

5 Evaluation

To evaluate the design described above we implemented a proof-of-concept as described in the previous chapter. This proof-of-concept provided information on performance and precision of the design. The following chapter describes the setup used to evaluate our design and presents our findings.

5.1 Evaluation Setup

This section describes the data used in the evaluation and how this data was gathered / generated.

5.1.1 Road Network

To test our prototype we used data provided by an existing iTransIT system. Information on the route 15 bus was used to rebuild the road network along the route including bus stops and bus time tables. The data initially provided was not sufficient as the number of lanes per link was not given, neither was data on allowed turning manoeuvres and bus lanes. Thus we had to collect this data and update the existing spatially enhanced database so a precise model of the environment could be build. Figure 16 gives an overview over this network. It consists of 100 junctions and 146 links with a total of 288 lanes, 36 of which are bus lanes. The marked spots represent the calibrators used to create the traffic in the network.

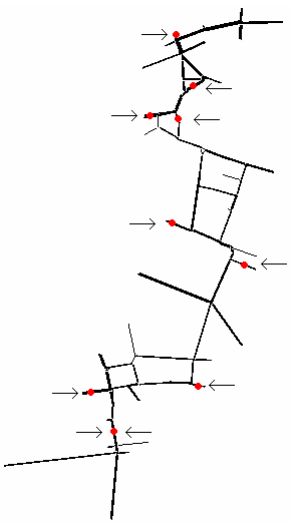

Figure 16, Map of the bus route 15

There is also network data available on the complete inner city of Dublin which spans about 15 km² and consists of 1305 junctions and 2822 links with a total length of approximately

307 km. However currently there is neither traffic data nor data on bus routes and stops available for this area so we only used the small section described above.

5.1.2 Traffic Patterns

Currently we do not have access to either historical or live traffic data. Therefore traffic pattern were generated to match empirical data taken along this bus route over the last months. The graphical version of the simulation provided a good tool to quickly judge traffic flow and density along the bus route. For our experiments we have identified 10 time periods during which traffic flows are either very low (1), low (2), medium (3) or high (4). The exact breakdown for a full day can be found in Figure 17.

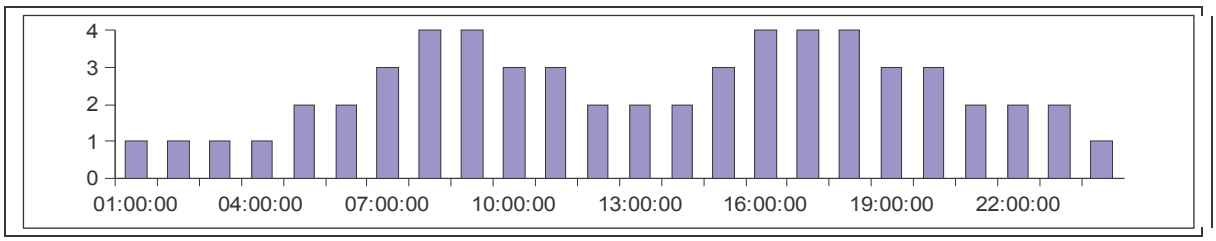

Figure 17, traffic levels over a full day

At first points along the route were picked where calibrators had to be positioned. They were chosen according to the bus route and the default turning ratios so that most vehicles would partially share the bus route and therefore impact the travel times of the bus. We found that 10 links with a total of 19 lanes and thus 19 calibrators were enough to generate a wide variety of traffic flows, from free flowing to congested. Once the positions were fixed different flow values, based on our empirical data, were tested to find those values generating a saturated to congested traffic flow. These values were then used to generate traffic during morning and afternoon peak hours. Further experiments were conducted to find reasonable values for the other traffic flows. We were able to retrieve acceptable results for the 'very low', the 'low' and the 'medium' state respectively based on our empirical data.

5.1.3 Bus Stop Times

As described in the section 3.2.3.2 the time spent by a bus at a bus stop varies depending on the number of passengers who wish to use a particular route and frequency of that service. However there is no data on bus utilisations and average passenger numbers available at the moment. Therefore we use average stop times rather than calculating these times based on average passenger numbers and bus stop intervals. Although this will not yield the most accurate results, it suffices to prove the general feasibility of this concept. More precise stop

times are only useful once access to live traffic data is available as the traffic situation has much larger impact on travel times than stop times.

5.1.4 Test System

All tests were run on a laptop computer with the following configuration:

- 2.5 GHz Intel Pentium 4 CPU
- 512 MB RAM
- Windows XP Professional
- Java 1.5.0_10
- MinGW toolset for *nix command line tools

Although the processes iTranSIM-DP, iTranSIM-DUS and SUMO are designed to be distributed over the network they were all run on this single machine because it reduced the administrative overhead of managing several machines.

5.2 Accuracy

To ensure the simulation provides accurate results at first the general simulation model, i.e. the car following and lane changing behaviour and its implementation in SUMO has to be examined. If this yields accurate results then the actual online simulation can be run. For the online simulation two aspects have to be analysed. First the accuracy of the online simulation needs to be evaluated and then the precision of the vehicle locations can be examined.

5.2.1 Offline Simulation

The accuracy of the car following model by Kraus and its usage in SUMO were evaluated previously [37,38]. The studies showed that SUMO can yield realistic results when it comes to microscopic traffic simulation. To verify this we ran several tests with different traffic levels, measuring the transit time of buses for the route 15. The results presented in Figure 18 show travel times for all vehicles of a route 15 inbound over the course of one day.

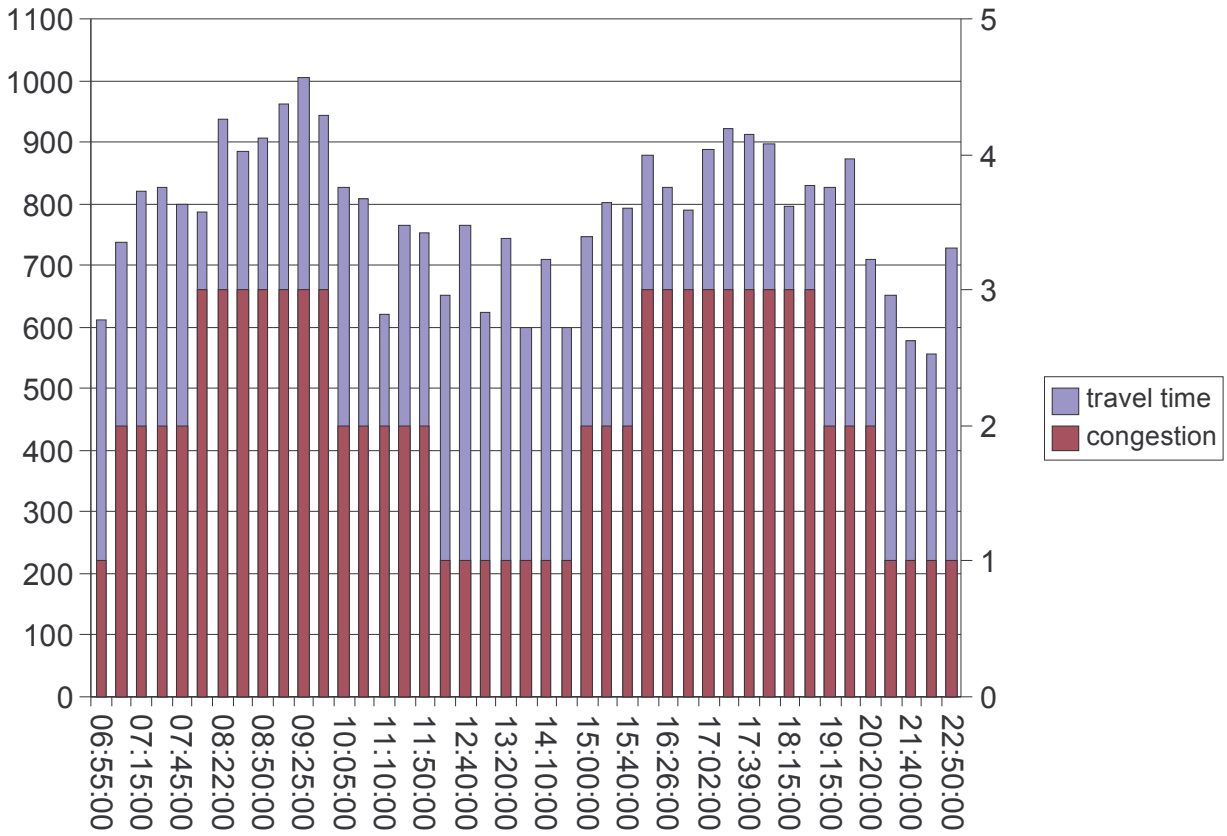

Figure 18, Simulated Bus Travel Times

The graph shows two important facts: Firstly the amount of traffic in the network influences the travel times. The more traffic exists in the system the longer it takes the bus to finish its trip. This is a result we expected and it shows that the relationship between traffic and travel times is modelled correctly by the simulator. The second finding is that although there was no live data available for this test the travel times are similar to empirical data taken for this route. A minimum of 10 minutes for this trip is a bit lower than our measurements that yielded 11 to 12 minutes as an absolute minimum, the maximum of approximately 17 minutes is a good average for medium congested roads although it was exceeded several times during our empirical tests, but this was due to very heavy congestion which only could be simulated realistically with live or at least historical traffic data.

To calibrate the traffic simulator the vehicle counts and mean vehicle speeds can be compared to real world data. The main parameters that could be adapted are those of the vehicles, i.e. acceleration, deceleration, probability of imperfect behaviour. Furthermore the mix of different vehicle types could be adjusted to yield accurate results with the simulation. To speed up the calibration Hourdakis et. al proposed a calibration methodology which includes the automation of parts of the simulation [39].

From these experiments and the studies mentioned above it can be concluded that the simulation does yield realistic result for the traffic simulation. To test the absolute precision of the system however it is necessary to gain access to live data and to perform online simulations based on this data, rather than on empirical data alone.

5.2.2 Online Simulation

To evaluate the accuracy of the online simulation it is necessary to get access to live traffic data. Although an integration of this data into the existing iTransIT system is planned the data was not yet available.

Once real vehicle count data becomes available the evaluation can be run using the implementation as previously described. Further changes to the prototype are not necessary as it already uses the vehicle counts from the spatial database. The only difference is that this data is then updated regularly whereas at the moment the values in the database do not change and are based on empirical data.

To assert the accuracy of the online simulation the flows and densities at the positions of the calibrators have to be compared to the real world values. Therefore an E1-detector can be used in the simulation, the definition of these detectors is already implemented in the iTranSIM-DP alongside with the generation of all other input files. In addition to that an iTransIT service has to be implemented to store all the changes in the real detector loop count data (i.e. a historical database) so that the values can be compared to the output of the E1-detectors. A special consideration should be given to the behaviour of the calibrators in the congested traffic state because here the same flow values apply as for the free flow state. An adjustment of the calibrators might become necessary to generate realistic results. The timing of calibrators can be adapted so they monitor a longer or shorter time frame before they add or remove vehicles. If a simple adjustment of the calibrators does not deliver acceptable results the algorithm used to adapt the vehicle flows in the simulation can be replaced by a more complex algorithm adapting the vehicle densities [30].

In addition to that more realistic results could be produced if the real traffic light sequences are used in the simulator. Currently the sequences are automatically generated by SUMO's netconvert, however once this information is available in the iTransIT system it could be incorporated into the simulation.

5.2.3 Vehicle Locations

With an accurate online simulation the bus travel times and positions can be measured and compared to the real bus travel times. Thus empirical data for real bus trips has to be gathered so it can be compared to the simulation results.

As currently only the vehicle counts are adapted dynamically the results of this evaluation will have quite a high error rate. To gain accurate results a stop time estimation has to be implemented based on the vehicle frequency and the passenger arrival rate. This requires arrival rate data for bus routes at all stops along the route, data that is currently not available in the iTransIT system.

For the beginning at least the travel times between stops can be analysed to see if these results are accurate.

Once this data becomes available the passenger service time for individual buses on specific bus routes can be calculated. This calculation could be accomplished by either a universal processor providing the information to the iTranSIM-DP or by a dedicated processor updating the data directly in the simulator. If the travel times between stops are accurate, then the stop time becomes the major source of inaccuracy. Thus the system to calculate the PSTs has to be calibrated to improve the results. The average values for dead time, passenger embark time and passenger disembark time could be adapted. As the data provided (passenger arrival rate / disembark ratio) to the system is most likely to be empirical data it has to be evaluated how accurate this data actually is.

5.3 Extensibility

Extensibility of the system is important so if new data becomes available it can be accommodated in the simulation.

The extensibility of the proposed implementation depends mainly on the available features in SUMO. If an aspect of the simulation can be addressed with SUMO, then the data only needs to be fetched and transformed into the proper input file. An example for such an aspect would be the traffic light sequences. As SUMO already supports the definition of specific sequences for particular traffic lights no changes to the simulator are necessary.

If something is not supported in the simulator, then in addition to processing the information in the iTranSIM system and providing it to SUMO, the functionality also has to be

implemented. For this project the calibrator was the biggest change we had to make to SUMO and it involved a substantial amount of design, programming and testing efforts.

5.4 Performance / Scalability

The performance of software systems is always a concern during evaluation. For this project however it is especially important because the system has to confirm to (soft) real-time requirements in order to be of any use. A real time system is capable of finishing its calculations before a given deadline. In the case of the iTranSIM / SUMO combination this means that a simulation step has to be completed in less than one second, so that the simulation can stay up-to-date. It is however a soft real time constraint as sporadic breaches of this requirement only deteriorate the results (vehicle positions might be outdated by a few seconds), but they do not invalidate them.

We have identified four possible bottlenecks that could slow down our system. Three of those are related to the communication between the components, the fourth one is the processing time needed for the simulation.

5.4.1 Spatial API

The first bottleneck to be investigated is the data retrieval using the spatial API. The retrieval of static information is not considered to be a bottleneck as it only happens once, before the simulation is actually started. The duration of this process can be incorporated in the simulation control so the simulation starts on time even if the building of the input files including the data retrieval using the spatial API took some time.

However the retrieval of the vehicle counts for the calibrators has to happen continuously to keep the calibrators up-to-date. Thus the more calibrators exist in the system the more requests to the spatial API have to be performed at each timestep. The objectIDs of the lanes with induction loops can be fetched before the simulation starts thus the `S_API::select()` method is only called once and not at every update. That leaves two calls to the spatial API per calibrator and timestep. `S_API::retrieve()` is first used to retrieve the loop count form "D_Lane" and the second time for the name of the link from "R_Link".

The time needed to retrieve the data for one calibrator using the spatial API aggregates to about 50 ms, which means in turn that even when all the other processes are neglected, only 20 calibrators can be retrieved in one second. With all the other processes getting their share

in the CPU and a reasonable safety margin this value is very likely to be closer to 10. Hence it must be concluded that the spatial API in its current form only allows very limited, small scale online simulations.

To continue the evaluation an alternative data retrieval method was implemented using JDBC for direct access to the spatial database. This brought down the time to retrieve detector loop counts to <10 ms, mainly because using SQL allows retrieving all loop counts at once, where the spatial API requires two requests per loop count. The SQL statement used to retrieve the count data is shown in Listing 9.

```
SELECT li.id, la.pos
FROM R_Link li, D_Lane la
WHERE la.detectorLoopCnt IS NOT NULL
      AND la.fk_link_autoId = li.autoId;
```

Listing 9, SQL Statement for Loop Count Retrieval

Further tests with the JDBC interface showed that the average time to retrieve count data for 1000 calibrators was just above 105 ms. Therefore this method could also be used for large scale simulations. Even if the whole inner city of Dublin with approximated 4000 lanes would be equipped with calibrators the time to retrieve the loop counts would be between 400 and 500 ms, thus leaving enough room for further expansion.

5.4.2 Simulation Control

The second possible bottleneck is the data transfer between iTranSIM and SUMO. This includes the Update Calibrator and the Simulate Steps command. As described in 4.3.3 the size of a Simulate Step command is 15 byte, the Update Calibrator message has a minimum size of bytes 23 byte (1 byte calibrator ID), but even with longer calibrator IDs the size should not exceed 30 byte.

In our setup the transfer for 19 calibrators accumulates to 585 byte, which can transferred in <1 ms over a socket on the loopback device or over a standard 100 MBit Ethernet connection. For the 4000 calibrators in Dublin inner city the total size of the calibrator data is about 120 kBytes (again, this is a hypothetical number, assuming every lane in the inner city is equipped with a detector loop). So even the data for 4000 induction loops can be transferred over the socket in <10 ms for loopback connections and in about 20 ms over 100 MBit

Ethernet (theoretically it would take 10 ms to transfer this data over fast Ethernet but due to the overhead of TCP and slow start the actual transfer time is significantly higher).

Compared to the time needed to retrieve the loop count data from the iTransIT system the time consumed by the simulation control is considerably lower and should not cause delays even for large simulations.

5.4.3 Output Processing

During the implementation it already became clear that using the network state dump as the source for vehicle positions is not feasible as it is just too much data to be transferred and parsed on time. The vehicle type probe was implemented to overcome this problem and supplies exactly the data needed.

Each entry in the output describes one vehicles location and is approximately 100 bytes long (depending on the length of vehicle ID, link ID and lane ID). In our experiment we had a maximum of 8 buses running at the same time (peak hours, 4 inbound and 4 outbound). Thus the total data transferred was less than 1 kilo byte. The data was then parsed by a customised SAX parser, the total time to transmit and parse the data was <10 ms.

To simulate larger networks tests were run with 1000 vehicle positions transmitted per timestep. This equals approximately the size of Dublin Bus' fleet, keeping in mind that not all vehicles are in use at the same time this provides a margin for further expansion. The average time needed to transmit and parse 1000 vehicle locations was approximately 480 ms, about 50 ms for the transmission and 430 ms for parsing the data. Thus a simulation of the complete bus fleet in Dublin would still be possible using this approach.

The update of the data gathered was accomplished using JDBC. In its current version the spatial API does not allow updates to location objects, therefore the data was written to the database using SQL statements and JDBC. The time needed for the data update was similar to the results in section 5.4.1 and did not cause a significant delay during our tests.

5.4.4 Simulation

Apart from the communication the actual computation of the simulation was identified as a possible bottleneck as it is very CPU intensive. Therefore several tests were run to calculate the average time needed to perform a simulation step.

The main influence on the time needed for the computation of the next step is the number of active vehicles in the simulation. The time can be approximated to be a linear function of the number of active vehicles, because for every vehicle the calculation of its new position is independent of the other calculations. Figure 19 shows the results for different amounts of active vehicles in the simulation. The measurement for 600MHz was taken on the same machine, using CPU throttling.

CPU	active Vehicles		
speed	50	400	700
2500 MHz	5 ms	13 ms	24 ms
600 MHz	24 ms	75 ms	132 ms

Figure 19, Simulation Performance Times

The data shows that firstly the small simulation can be run without any problems even on a relatively slow machine, and secondly that the CPU time needed for calculation grows in what appears to be a linear manner. This coincides with the assumptions made earlier that the vehicles are the main influence on the processing time and that they are independent.

To extrapolate these values to a full scale simulation the maximal amount of vehicles in the simulation is necessary. From the data available we calculated the total length of the road network in Dublin's inner city which adds up to approximately 307 km. According to Hoogendoorn et al. [23] the maximal number of vehicles per km route is about 165. This means that all cars are standing 'bumper to bumper' and not moving at all. In that case Dublin's inner city has a capacity of about 50000 vehicles. However in reality this value will never be reached as it would mean that every road is completely blocked. From that data we induced that about 20000 active vehicles would be a realistic value for a heavily congested traffic state. Extrapolating the time taken for the small simulation this would mean that the calculation of a single timestep would take about 700 ms, which is still a feasible value.

5.4.5 Conclusion

Summarizing the finding of this section it can be said that apart from the slow performance of the spatial API all other components proofed to run the online simulation on the small test network within a fraction of the time available. Further analysis showed that running a full scale simulation of Dublin's inner city can still be feasible.

Although a full scale simulation of Dublin's inner city is probably the maximum of what can be achieved on a single CPU PC, there is potential for distribution and further enhancements of the components. The 3 main processes (iTranSIM-DP, iTranSIM-DUS, SUMO) can be

distributed on dedicated machines. To speed the output processing up the vehicle type probe could be implemented to use a binary output format rather than XML which would reduce the amount of data transmitted and the time required for parsing the data.

However one issue that needs to be addressed is the poor performance of the spatial API. Due to a short timeframe we were not able to investigate the problems any further, however this problem was previously reported and a revised version of the spatial API is in planning. To comply with the iTransIT framework it would be an advantage if the data retrieval could actually be accomplished using the spatial API because several processes that access the spatial database directly might cause inconsistency in the datasets.

5.5 Evaluation Conclusions

The Evaluation showed that it is possible to build an online simulation based on the iTransIT framework and the results yielded so for are very promising as to how accurate such a system can be. In addition to the general feasibility the extensibility and the performance were analysed.

The results showed that the extension of the current model is easy because new data can be modelled according to the iTransIT specification. However extending the implementation might prove more complicated if a change of the simulation engine is necessary.

The performance analysis provided results to prove that online simulations even of large urban environments are feasible and can be run on standard PC hardware.

6 Conclusion

The goal of this project was twofold:

1. Examine the possibility of creating an online traffic simulation using ITS data while complying with the iTransIT framework.
2. Evaluate the accuracy of such a simulation and examine the feasibility of using such a system for automatic vehicle location.

We were able to implement an online simulation using the iTransIT framework. The iTransIT infrastructure and the spatial API provided a convenient way to access existing data and to populate our data model. Even though there was an issue regarding the performance of the spatial API this cannot be seen as a general design problem rather than an implementation issue which should be fixable. It can therefore be concluded that the iTransIT architecture does provide a framework for building an online simulation and subsequently an AVL system based on that simulation. Further analysis of the design showed that the simulation in itself produced feasible results and that the system is capable of running large scale simulations even on normal PC hardware.

The second goal, the analysis of the accuracy could not be achieved due to the fact that an integral part of the necessary infrastructure is not yet available: the live traffic data. However the system was designed in a way that as soon as this data becomes available the evaluation can be carried out. A detailed description of how this evaluation should be performed was also provided.

Although the goals were not achieved completely, the results of the analysis are very promising. We were able to prove that a simulation based AVL system can be implemented and as soon as traffic data becomes available the accuracy of the system can be evaluated.

This work can be seen as a first step towards an advanced public transport system providing real time passenger information to the passengers of Dublin Bus. Such a system could improve drastically upon the situation in Dublin [13]:

> *Passengers having only very broad indications of the times that buses will pass intermediate stops, which in turn leads to perceived unreliability and a need to allow longer than necessary waiting times at stops.*

A proper APTS would raise the passenger's confidence in the public transport therefore raising the number of passengers and subsequently lowering the amount of private traffic in Dublin, helping to create a less congested and cleaner city.

6.1 Future Work

The work presented in this thesis can serve as the basis for further evaluation of simulation based AVL. The main focus currently lies on further evaluation of the system but other projects related to ITS and iTransIT could also benefit from this project. This section summarises where research could go from here.

6.2 Different Simulation Models

Different simulation models could be implemented and compared provided an accurate online simulation is generally possible. Currently the car following and lane changing model by Gipps and Kraus is used (see section 2.2.4.1 for details). Other models such as cellular automata might provide similar or better results with a smaller need for processing power.

To test the other models an appropriate simulator implementing the model is necessary. Instead of implementing new simulators from scratch, SUMO might be used to provide the basis for the new models. This would also allow reducing the adaptation of the iTranSIM system to a minimum. Changing the simulator however would make major changes to the iTranSIM system necessary. As most of the components in the iTranSIM system deal either with exchanging data or control messages with the simulator all these components need to be adapted to the new simulator.

The evaluation of simulation models could be used to find the optimal model for this system, but it could also be used to compare the models with respect to a feature not yet described in the literature.

6.2.1 Alternative Applications for Online Simulations

In this project the main purpose of the online simulation was to provide an environment for tracking vehicles. However other possible applications of the simulation are also possible provided the online simulation proves to be accurate enough.

Congestion Level Prediction

One application where online simulations were used before is the field of congestion level prediction. For lanes that are equipped with detector loops the congestion levels can be calculated based on the flow and density data provided by these devices. For roads without any detector loops build into them there in no direct way to compute the congestion level. Therefore an online simulation could provide a basis for gathering congestion levels not only for roads equipped with induction loops but for the whole network. However to generate an online simulation accurate enough for such an approach it would necessary to have a high number of calibrators spread homogeneously over the network.

A congestion level prediction based on online simulation was previously implemented and described by Schreckenberg et. al in 2001 [30]. For this project a simulator based on cellular automata was used to model the main road network of the city of Duisburg, Germany with a total length of 165 km and over 730 check points (calibrators). Although they describe their results as "sufficient to serve as a basis for further investigation" a detailed analysis of the system's accuracy is not given.

VANET Research

The field of vehicular ad-hoc networks (VANETs) explores the possibilities and limitations of mobile inter vehicle communication. Traffic simulations can be used to test different setups and different protocols.

The Distributed Systems Group at Trinity College currently runs several projects related to VANETs. Therefore the online simulation of Dublin might provide a suitable testing environment for new ideas or protocols. To support network research several tools were developed to connect the SUMO traffic simulation with network simulators such as ns-2. One example is the MOVE project by Feliz Karnadi et. al [40]:

> *The output of MOVE is a mobility trace file that contains information of realistic vehicle movements which can be immediately used by popular simulation tools such as ns-2 or qualnet.*

Furthermore other parties have implemented additional features to support VANET research. The itm remoteserver that was used in this project to update vehicle count values and to send control commands was originally intended to enable VANET applications to interact with the simulation [36]:

> *To achieve a realistic simulation setup, an external road traffic simulator is used in our setup. [...] The network simulator and mobility generator are connected online by the proposed protocol. Thus, the mobility behaviour of each single network node can be influenced by the VANET application running inside the network simulator.*

Thus the online simulation of Dublin could provide the researchers at Trinity College with a realistic simulation environment for their applications.

References

[1] Bus Átha Cliath – Annual Report and Financial Statements 2006, Dublin, Ireland, 2007

[2] Karbassi, A., Barth, M., "Vehicle route prediction and time of arrival estimation techniques for improved transportation system management" in *Proceedings of the IEEE Intelligent Vehicles Symposium, 2003,* pp. 511- 516, 9-11 June 2003

[3] McGreevy, R., "*Call for radical review as buses slow to a crawl*", The Irish Times, Dublin, Ireland, 09 June 2007

[4] Figueiredo, L., Jesus, I., Machado, J.A.T., Ferreira, J.R., Martins de Carvalho, J.L., "Towards the development of intelligent transportation systems", in *Proceedings of the IEEE Intelligent Transportation Systems Conference 2001* (ITSC'01), pp. 1206-1211, Oakland, USA, 25 – 29 August 2001

[5] McQueen, B., McQueen J., "*Intelligent Transportation Systems Architectures*", Boston, USA, Artech House Books, 1999, ISBN 089006525X

[6] Brennan, S., "*Global Smart Spaces – The Smart Traveller Information System*", M.Sc. Thesis, University of Dublin, Trinity College, Dublin, Ireland, September 2006

[7] Hwang, M., Kemp, J., Lerner-Lam, E., Neuerburg, N., Okunieff, P., "*Advanced Public Transportation Systems: State of the Art Update 2006*", New York, USA, 2006

[8] Riter, S., McCoy, J. "Automatic vehicle location - An overview" *in IEEE Transactions on Vehicular Technology*. Vol. 26, no. 1, pp. 7-11., February 1977

[9] Zhao, L., Ochieng, W.Y., Quddus, M.A., Noland, R.B., "An Extended Kalman Filter Algorithm for Integrating GPS and Low Cost Dead Reckoning System Data for Vehicle Performance and Emissions Monitoring" in *Journal of Navigation* (2003), 56, pp. 257-275, Cambridge University Press, Cambridge, UK, May 2003

[10] Shyang-Lih, C., Li-Shien, C., Yun-Chung, C., Sei-Wan, C., "Automatic license plate recognition" , *in IEEE Transactions on Intelligent Transportation Systems*, Volume: 5, Issue: 1, pp. 42-53, March 2004

[11] Accurate vehicular positioning using a DAB-GSM hybrid system Rooney, S., Chippendale, P., Choony, R., Le Roux, C., Honary, B., "Accurate vehicular positioning using a DAB-GSM hybrid system", in *Proceedings of the IEEE Conference on Vehicular Technology 2000* (VTC2000-Spring), pp. 97-101, Tokyo, Japan, 15 – 18 May 2000

[12] Caulfield, B., O'Mahony, M. M., "The provision of on street passenger information via real time passenger information; A case study of Dublin", in *Proceedings of the IEE Conference on Road Transport Information & Control 2004* (RTIC2004), pp. 1-10, London, UK, 20 - 22 April 2004

[13] Dublin Bus, *Dublin Bus Network Review 2006*, Dublin, Ireland, February 2006

[14] *Satellite device to tell you where that bus got to*, Irish independent, 21 April 2006

[15] Dublin Bus, *Real Time Passenger Information Ssystem*, [cited August 2006], Available from: http://www.dublinbus.ie/projects/real_time_passenger_information_system.asp

[16] Meier, R., Harrington, A., Cahill, V., "A Framework for Integrating Existing and Novel Intelligent Transportation Systems," in P*roceedings of the 8th International IEEE Conference on Intelligent Transportation Systems* (IEEE ITSC´05). Vienna, Austria: IEEE Computer Society, 2005, pp. 650-655.

[17] Meier, R., Harrington, A., Cahill, V., "A Distributed Framework for Intelligent Transportation Systems," in *Proceedings of 12th World Congress on Intelligent Transport Systems* (ITSWC2005). San Francisco, California, USA, 2005.

[18] Meier, R., Harrington, A., Termin, T., Cahill, V., "A Spatial Programming Model for Real Global Smart Space Applications," in *Proceedings of the 6th IFIP International Conference on Distributed Applications and Interoperable Systems* (DAIS 06), LNCS 4025. Bologna, Italy: Springer-Verlag, 2006, pp. 16-31.

[19] Meier, R., Harrington, A., Cahill, V., "Towards Delivering Context-Aware Transportation User Services," in *Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems* (IEEE ITSC 2006). Toronto, Canada: IEEE Computer Society, 2006, pp. 369 – 376.

[20] Lee, D., Meier, R., "Primary-Context Model and Ontology: A Combined Approach for Pervasive Transportation Services," in *Proceedings of the First IEEE International Workshop on Pervasive Transportation Systems* (IEEE PerTrans 2007). White Plains, New York, USA: IEEE Computer Society, 2007, pp. 419-424.

[21] Maerivoet, S., De Moor, B., "*Transportation Planning and Traffic Flow Models*", Technical Report, Katholieke Universiteit Leuven, Belgium, July 2005

[22] Lighthill, M.J., Whitham, G.B., "On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads", in *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Vol. 229, No. 1178 (May 10, 1955), pp. 317-345, London, UK

[23] Hoogendoorn, S.P., Bovy, P.H.L., "State-of-the-art of vehicular traffic flow modelling" in *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, Volume 215, Number 4 / 2001, pp. 283-303

[24] Krauß, S., "*Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics*", Doctoral Thesis, University of Cologne, Germany, August 1998

[25] Van Aerde, M., Hellinga, B., Baker, M., Rakha, H., "I*NTEGRATION: An Overview of Traffic Simulation Features*", Transportation Research Board Annual Meeting Washington, D.C., USA 1996

[26] Esser, J., Schreckenberg, M., "Microscopic simulation of urban traffic based on cellular automata" in *International Journal of Modern Physics C,* Vol. 8, No. 5 (1997), World Scientific Publishing Company, 1997

[27] Algers, S., Bernauer, E., Boero, M., Breheret, L., Di Taranto, C., Dougherty, M., Fox, K., Gabard, J.F., "*SMARTEST - Review of Micro-Simulation Models*", SMARTEST Project Deliverable D3, August 1997

[28] Van Aerde, M., and Rakha, H., "Multivariate Calibration of Single Regime Speed-Flow-Density Relationships", in *Proceedings to the VNIS/Pacific Rim Conference*, Seattle, WA, USA, 1995, pp. 334-341

[29] Brockfeld, E., Kühne1, R.D., Skabardonis, A., Wagner, P.,"Toward Benchmarking of Microscopic Traffic Flow Models" in *Transportation Research Record,* Volume 1852 / 2003, Transportation Research Board of the National Academies, 2003, pp. 124-129

[30] Wahle, J., Neubert, L., Esser, J., Schreckenberg, M., " A cellular automaton traffic flow model for online simulation of traffic", in *Parallel Computing*, Volume 27, Number 5, April 2001 , pp. 719-735

[31] Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P., "SUMO (Simulation of Urban MObility); An open-source traffic simulation" in *~Al-Akaidi, A. [Ed.]: Proceedings of the 4th Middle East Symposium on Simulation and Modelling* (MESM2002), SCS European Publishing House, pp. 183-187, Sharjah / United Arab Emirates, Sept. 2002

[32] Gipps, P.G., "Behavioral Car-Following Model for Computer Simulation" in *TRANSPORT. RES.* Vol. 15B, no. 2, pp. 105-111. 1981

[33] Krajzewicz, D., Rössel, C., "*SUMO - Simulation of Urban Mobility - User Manual*", Revision: 3977, 2007

[34] Fernandez, R., Tyler, N., "Effect of Passenger – Bus – Traffic Interaction on Bus Stop Opperations", in *Transportation Planning and Technology*, Vol. 28, No. 4, August 2005, pp. 273-292

[35] Noland, R.B., Polak, J.W., "Travel time variability: a review of theoretical and empirical issues" in *Transport Reviews*, Vol. 22, No. 1, 2002, pp. 39-54

[36] Wegener, A., "*A Mobility Interface Protocol to connect Ad Hoc Network Simulations to Realistic Mobility*", August 2007

[37] Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P., "An Example of Microscopic Car Models Validation using the Open Source Traffic Simulation SUMO" in *Proceedings of the 14th European Simulation Symposium,* Dresden, Germany, October 2002

[38] Krajzewicz, D., Hartinger, M., Hertkorn, G., Nicolay, E., Rössel, C., Ringel, J., Wagner, P., "Recent Extensions to the open source Traffic Simulation SUMO" in *Proceedings of the 10th World Conference on Transport Research*, WCTR04 - 10th World Conference on Transport Research, Istanbul, Turkey, July 2004

[39] Hourdakis, J., Michalopoulus, P.G., Kottommannit, J., "*A Practical Procedure For Calibrating Microscopic Simulation Models*", Transportation Research Board Annual Meeting Washington, D.C., USA, 2003

[40] Karnadi, F.K., Mo, Z.H., Lan, K.C., "Rapid Generation of Realistic Mobility Models for VANET", in P*roceedings to the IEEE Conference on Wireless Communications and Networking , 2007* (WCNC 2007), Kowloon, China, March 2007