Microscopic Online Simulation for Real time Traffic Management

Marc Philipp Miska

(2007)

Cover illustration: Marc Philipp Miska

Microscopic Online Simulation for Real time Traffic Management

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft
op gezag van de Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 30 januari 2007 te 10.00 uur
door

Marc Philipp Miska

Diplom Ingenieur (University of Hannover, Germany)
geboren te Bitburg, Germany

Dit proefschrift is goedgekeurd door de promoter:
Prof. dr. H.J. van Zuylen

Samenstelling Promotiecommissie

Rector Magnificus	Voorzitter
Prof.dr. H.J. van Zuylen	Technische Universiteit Delft, promotor
Prof.dr.ir. P.H.L. Bovy	Technische Universiteit Delft
Prof.dr.ir. S.P. Hoogendoorn	Technische Universiteit Delft
Dr.ir. H. van Lint	Technische Universiteit Delft
Prof. Dr. J. Barceló	Technical University of Catalonia, Spain
Prof. Dr. M. Kuwahara	Tokyo University, Japan
Prof. Dr. H. Mahmassani	University of Maryland, USA

TRAIL Thesis Series T2007/1, The Netherlands TRAIL Research School

Published and distributed by:
TRAIL Research School
P.O. Box 5017
2600 GA Delft
The Netherlands
Phone : + 31 (0) 15 27 86046
Fax : + 31 (0) 15 27 84333
E-mail : Info@rsTRAIL.nl

ISBN-10: 90-5584-082-3
ISBN-13: 978-90-5584-082-3

Keywords: microscopic, online, simulation

Printed in the Netherlands

心から愛するけいへ

Preface

The work reported in this thesis was supervised by Professor Dr. Henk J. van Zuylen as promoter and ir. Theo H.J. Muller as daily supervisor at the Transport and Planning Department of the Delft University of Technology. I am grateful to both, Henk and Theo, for the discussions we had, the things I have learned from them and the faith they had in my work during the four years of my research. Their supervision of my work gave me the freedom to follow my own ideas but they also managed to make sure that I did not get lost in those.

Next I would like to acknowledge the financial support of the AVV Transport Research Center of the Dutch Ministry of Transport, Public Works and Water management, which made this thesis and the underlying research possible in their Regiolab project. Further thanks go to the TRAIL research school which has done a very well job in supporting me during the PhD research and their courses broadened my mind in various directions.

I would like to thank Professor Piet Bovy, who encouraged me continuously in my research and for initializing my internship at the Royal Netherlands Embassy in Tokyo, Japan. As it turned out, this connection was very fruitful and was the foundation for my further working life. At the Embassy, I would like to give my special thanks to the Office of Science and Technology. Philip Wijers, Erik Blomjous, Rob Stroeks, Thomas Bleeker, and Mihoko Ishii, who took good care of me during my stay and in their limited free time, introduced me to life in Japan. Thanks to them I had the opportunity to work with Professor Masao Kuwahara at the University of Tokyo, who is always able to ask the right questions in the right time to keep researchers hungry for knowledge and pushing themselves further. A special thanks therefore to Professor Kuwahara and the members of his laboratory and especially to Professor Edward Chung, for his interest in my work.

At the Department of Transport and Planning I want to give special thanks to my fellow PhD students Minwei Li, Dong Ngoduy, Francesco Viti, Thomas Dijker, Frank Zuurbier and Dusica Joksimovic. They were a great support during my entire research period and always able to help me out with my work and private issues. Further I would like to mention Dr. Hans van Lint, who with his frank way made my life at the institute always a pleasure.

I want to thank my parents, who invested a lot in my education and who supported all my decisions in life, even though they might did not agree to a fully extend. Without them this thesis would not be possible in the first place.

Table of Contents

List of Figures

1 Introduction

The mobility demand of our society increased rapidly in the last decades. To ensure accessibility the traffic infrastructure needs to be used more efficiently. One way to achieve this is the introduction of dynamic traffic management (DTM).
DTM monitors the traffic situation and tries to optimize it by applying different control measures like speed limits, route guidance, lane closure, ramp metering, intersection control or others to the traffic network. In its ideal form it is a continuous cycle that gets evaluated by checking the effects of the taken measures to the traffic situation, although in practice the measures are often predefined and not tuned to the detailed actual traffic situation. Traffic models are used to support traffic engineers with the optimization task by predicting the effect of different measures before applying them to the real network. An illustration of this DTM cycle can be found in Figure 1. Modeling traffic for optimization purposes can be done in several ways. However, this work focuses on a framework for online simulation for predicting the influence of traffic control measures to the network.

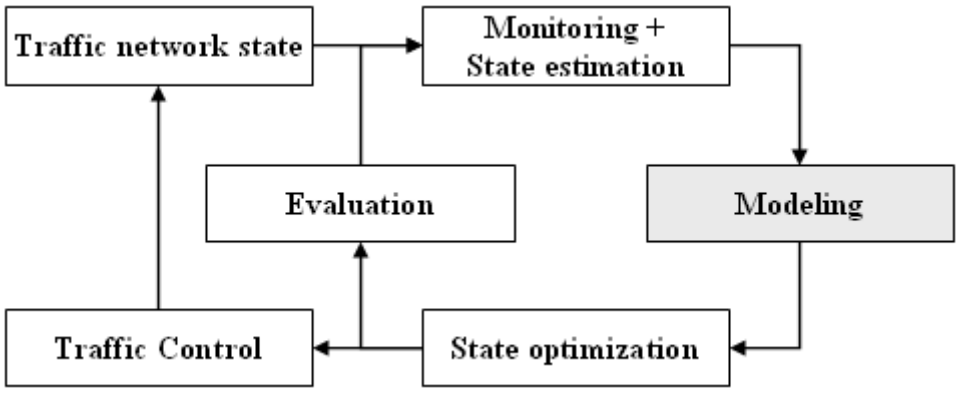

Figure 1: DTM cycle for optimizing traffic networks

Online simulations of traffic networks require a whole suite of applications, from data gathering, cleaning and fusion, over origin/destination (OD) estimation and prediction, to the simulation itself, including route choice, driving behavior and travel behavior. Thus, online simulation is based on several methods and models developed in several disciplines like control, mathematics, human behavior, and vehicle dynamics.

1.1 Problem description

Therefore, as shown above a system for online traffic management needs to be consisting of several tools to perform the wanted task. In 2004, a report examined the requirements for a traffic control and information system in collaboration with traffic control centers and software designers (Heusch & Boesefeldt, 2004). The requirements they found were:

- Incorporation of roadside controllers and roadside equipment to the system,
- System distribution in main and regional control centers
- Monitoring day to day network operation
- Remote programming of roadside controllers
- Data accessibility throughout the system
- Message exchange between combined control centers
- User and permission control
- Archiving and statistical analysis of traffic data
- Data fusion
- Traffic state estimation and prediction possibilities

Users of such a system prefer a single application for the whole process over different tools, while the system itself should be flexible to cover a wide range of applications and further should be distributable over several locations. This not only challenges the design of such a system, but also raises the need for a product to cover a wide field of applications instead of specialization in topics like data handling, or simulation.
To tackle this problem, this thesis describes a system design that allows users to work with a single system, and to integrate state of the art tools for the distinct tasks of the process of traffic management.

1.2 Research approach

Goal of the research is to develop a modular microscopic online simulator in a scaleable framework for dynamic traffic management to allow researchers to test and benchmark newly developed algorithms in a uniform environment. In traffic control centers it can support traffic engineers in the network optimization task and allows them to update their system with latest developments without reengineering.
While in each field of online traffic management specialists are working on extensions and improvements on existing models or develop new ones, there is a lack

of research for combining these state of the art components to realize an online working system that meets the needs of the user and allows them to work with cutting edge technology as soon as it is available.

The research approach for this thesis can be divided in two major parts. The first part focuses on the design of a robust, flexible and scaleable framework for online traffic management, while the second part deals with the development and integration of a prototype online simulator to the framework and the application to a test area.

To meet the needs of developing a robust, flexible and scaleable system, a multi-agent approach has been chosen. Several agents, performing the tasks of data collection, data handling and communication, are distributed in a computer network. This allows an easy scalability of the systems in terms of computation power. For flexibility, all incorporated parts of the system are strictly autonomous and modular based, so that they can be exchanged if necessary due to personal needs or interests, while the synchronization is handled by the framework. To ensure the robustness of the system it includes methods to recover from technical failures of the computer network.

The simulator itself, called MiOS (Microscopic Online Simulator), consists of 6 different modules, which work independently and get synchronized by the multi-agent system:

- an online data interface,
- a cellular automata (CA) simulator for heterogeneous traffic,
- a route choice model,
- a driving behavior model based on microscopic observations,
- an OD estimation and prediction tools,
- and a postprocessor.

All parts are implemented in a prototype and calibrated with data from the real life laboratory in the region of Delft (Regiolab Delft) in the Netherlands. Regiolab Delft is a combined research effort of the Delft University of Technology, the research school TRAIL, the Dutch Ministry of Transport, the province of South Holland, the municipality of Delft and the traffic industry (Muller et al., 2002).

The monitored area (see Figure 2) includes a dense network of double detector loops on motorways (about every 500m), single loops urban roads and camera detection points in the city center of Delft, and radar detectors on the provincial roads.

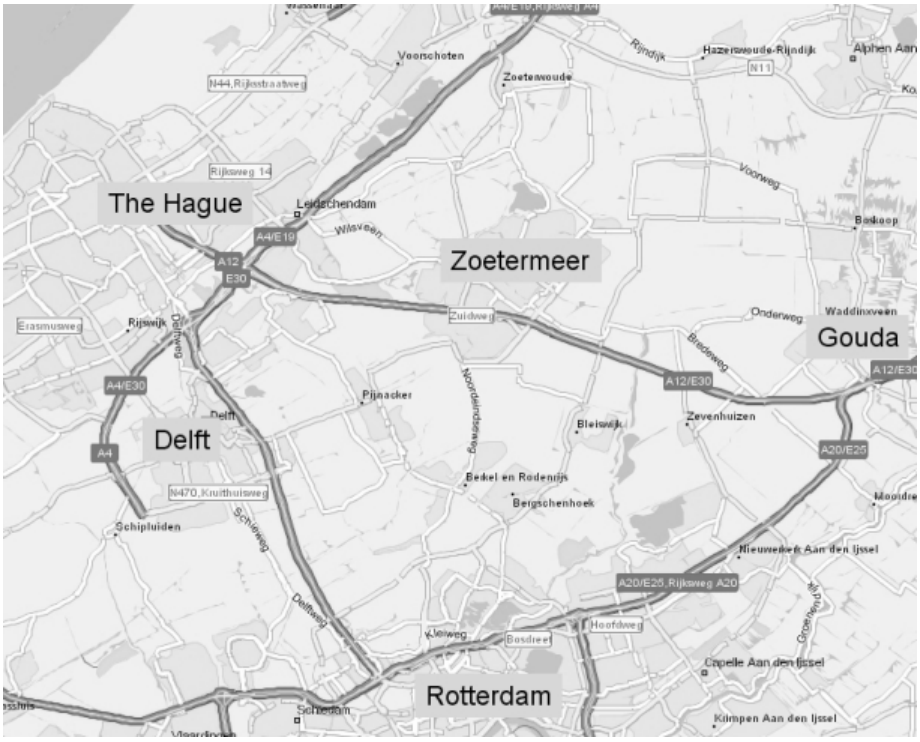

Figure 2: Region of Regiolab Delft (source Map24/TeleAtlas)

Information on flow rates, speeds, density and travel time are updated every minute and therewith an ideal environment for calibration and testing. Additional to pure traffic measurement, weather information for the city of Delft is provided by weather stations at the camera positions.

1.3 Scientific and technical contributions and practical relevance

This thesis makes the following main contributions to the subject of online traffic simulation and real-time traffic management:

1. A new driving behavior model based on belief networks with additional decision nodes has been developed that allows calibration with microscopic measurements
2. A cellular automata model for traffic simulations with a refined cell grid and shortened time step to enable a better representation of speeds and gaps
3. An extendable and scaleable framework for online traffic management is presented with a high degree of flexibility for online simulations

4. A robust prototype of a microscopic online simulator, developed in the described framework
5. Through its module based design, the framework provides an easy to use test bed for new components under stable conditions without further work on the framework

The practical relevance of this thesis is the development of MiOS as a support for decision making in traffic control centers. The robust design and high flexibility allow an easy integration to given environments in terms of hardware and software tools.

1.4 Thesis Outline

Following this introduction the thesis proceeds with the design and implementation of the framework for online traffic management (FrOnT) in Chapter 2. It describes the way of data gathering, storing, handling and its communication throughout the system. Subsequently, Chapter 3 gives an assessment of available online simulation models to determine the capabilities a basic online simulation model should have to meet the needs of researchers and practitioners. Chapter 4 then introduces the methodology of MiOS, including its modules:

- simulator,
- route choice,
- OD-estimation,
- traffic control, and
- driving behavior,

before Chapter 5 deals with the implementation of MiOS. The implemented simulator in the FrOnT system is tested in Chapter 6 with real world data from the Regiolab Delft. At the end of the thesis conclusions are drawn and suggestions for further research are given.

2 FrOnT – Framework for Online Traffic Management

2.1 Introduction

In Chapter 1 we listed the requirements of an online traffic management system, which now are explored in detail to define the requirements for designing an operational prototype of such a system.

Incorporation of roadside equipment and controllers, as well as the distribution of the system to main and regional control centers are key requirements mentioned in Chapter 1. Combined with the necessity of information interchange and remote control of roadside equipment this leaves the basic design choice to a distributed system with autonomous working elements that continuously communicate with each other. A powerful technique to accomplish this is the usage of a multi agent system, where several agents work together and communicated via the network structure (e.g. Internet). A further advantage of such a distributed system is the avoidance of redundant data. Information is not stored in every location, but requested on demand. Thus, archiving and statistical analyses do not require additional database capacities, but can use the collected and stored data from its origin. Further requirements are the possibilities for data fusion and traffic state estimation and prediction. These are independent from the system design and can be covered by single applications, running at the local control centers. However, to allow a co-operation between the local control centers and their co-operation with the main control center requires a constant data exchange which needs to be taken into account.

The framework for online traffic management (FrOnT) is the first contribution of this thesis. It is developed to meet the needs of the above described requirements, and additionally is able to combine state of the art research products for the best possible system for the individual user. FrOnT is an environment where real-time traffic data can be collected, stored and processed. Data from various sources get collected, unified with an internal structure and stored in a database. The data handling is performed by an agent society, which next to the data collection offers the possibility to trigger algorithms and software modules for data processing. These are independent from programming language or physical location and just need to be reachable via the computer network. Scalability is ensured within FrOnT by designing the system as a

docking unit. Every unit can be coupled with another unit and so FrOnT can grow with the computational demand to the system.

The following describes the design steps of FrOnT from the data collection, storage and handling to the decision making process for the implementation environment. FrOnT is an abstract framework, but during the design stages the connection to the DTM cycle described in Chapter 1 will be illustrated for a better understanding of the design decisions made.

2.2 Input data for online traffic management

Online traffic management relies on local real-time traffic information data to estimate the actual network-wide traffic situation in the network. Sources are (Westerman, 1995) among others:

- **Roadside detection**

  - *Infrared detectors*, which detect passing vehicles when a beam of light is interrupted. Active infrared detectors are additional able to recognize temperature differences (engine heat, body warms). Usually gathered information: aggregated flows and aggregated speeds in one minute intervals.

  - *Radar detectors*, which measure the presence and the speed of vehicles using the Doppler Effect. Usually gathered information: aggregated flows and aggregated speeds in one minute intervals. Further they can measure the height of the passing vehicle.

  - *Ultrasonic detectors*, which transmit ultrasonic sound waves instead of electro-magnetic radar waves. Usually gathered information: aggregated flows in one minute intervals plus a record of vehicle types, distinguished by their height.

  - *Induction loop detectors*, which detect vehicles entering a created electro-magnetic field by induction of Foucault currents. With two induction loops placed closely together (commonly 1 meter apart) not only the vehicle but also its speed can be detected. Usually gathered information: aggregated

flows and aggregated speeds in one minute intervals. Vehicle types can be obtained by induction patterns and is experimentally in use.

- *Video cameras*, which detect vehicles when entering and exiting a road stretch. Usually gathered information: aggregated flows and aggregated speeds in one minute intervals plus individual travel time data if a license plate recognition is used.

➢ **Floating car data**

- *Probe vehicles* transmitting traffic messages containing location, speed, and others at regular time intervals.

- *GSM data*, which is recently used to gather travel time data of the road network.

➢ **Others**

- *Historical data*, which gives a basic idea of the development of the traffic situation in previous times.

- *Weather conditions*, which have an influence on the traffic flow.

An additional input is the network description, including the network structure, road works and other information. The more different the sources of data are, the more different their way of storage, time lag, and accessibility. That raises the need of developing an internal structure, common for the different data sources, to process the information in a uniform way.

2.3 Internal FrOnT data structure

2.3.1 The entities of a physical network

The data structure of FrOnT is based on the given situation in the real world. The physical network, if looked at it from a schematic view (see Figure 3), consists of the road network with links and nodes, traffic controllers, measurement points and the vehicles in the network.

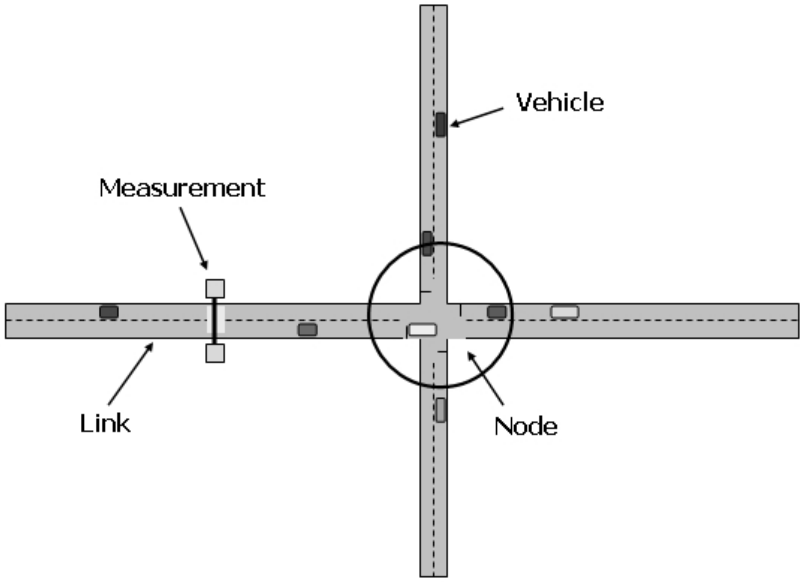

Figure 3: Scheme of the physical network

With these network entities found, it is possible to translate them into a class diagram (see Figure 4) based on the unified modeling language (Oestereich, 2002).

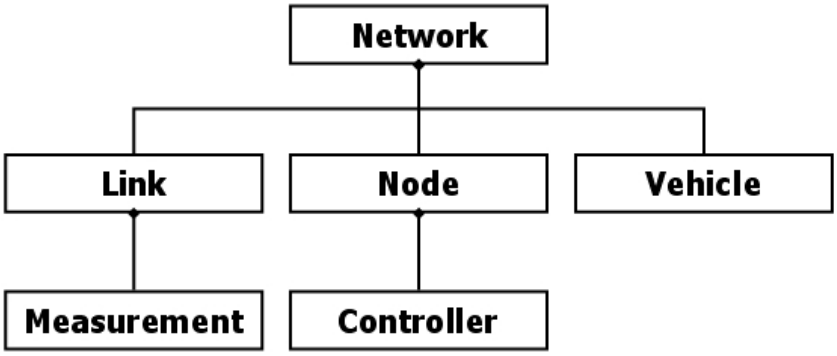

Figure 4: Class diagram to represent the physical network

After this first step of modeling, each entity needs to be described for itself. This means, that for each class in the diagram the attributes (describing variables) and methods (functions to change the values of the attributes) must be defined, which then leads to a more refined class model. The refinement starts from the bottom to the top, since the upper classes in the diagram have aggregations of the lower ones. In the following the data structure of FrOnT is described in entities. These are data packages

that can be handled, thus this part is still independent from further operation and is just the common ground for the actual data processing.

2.3.2 FrOnT - Controller entity

A controller in FrOnT is an abstract term for roadside elements for traffic control that can be influenced remotely, following a control program or displaying user defined information. There are various controllers in traffic engineering, like traffic lights, variable message signs (VMSs), parking guidance systems, or dynamic route information panels (DRIPs). To deal with several different controllers in a uniform data structure the FrOnT – Controller class is defined as an abstract class. An abstract class represents an abstract concept that allows defining common attributes and methods, without the possibility to create an instance of it. For further information on object oriented programming see Cox (1987). So later on a VMS, traffic light or DRIP is created, but all can be handled in FrOnT as a controller.
Common for all controllers are the attributes for the identification and the location of the controller in the network and the methods to set or read the values of the attributes. That allows changing the settings of all controllers during runtime. With FrOnT – Controller as an abstract parent class, the children classes VMS, DRIP and TrafficLight inherit all methods and attributes, but extend them with more specific details:

VMS:
Variable message signs can display warnings or restrictions to the road users. So, further attributes for this class are the displayed information as well as its effect to the network, like lane closure, maximum speed and others.

DRIP:
Dynamic route information panels have also a field for the displayed information, but instead of the network, they have to store the effects to the route choice in the network.

TrafficLight:
Traffic lights use values for the cycle time, the duration of the green, red and amber phase, as well as the off-set of the cycle to allow a fixed time traffic light control.

If further control mechanisms are added in future, the usage of the abstract class FrOnT – Controller allows the implementation without further changes in the other program parts.

2.3.3 FrOnT - Measurement entity

To model the various measurement installations in the network an abstract class FrOnT – Measurement is used. Even though different values can be measured with different methods, the abstract class holds the whole variety and defines several different methods of data gathering. The usage of an abstract class however, gives the possibility to interpret the data individually. In FrOnT – Measurement the processed values of the raw data are stored. The attributes of a FrOnT – Measurement are:

- Timestamp of the measurement
- Speed (individual vehicle / aggregated value)
- Flow (individual vehicle / aggregated value)
- Vehicle type (if available)
- Height of the vehicle (if available)
- Width of the vehicle (if available)
- Condition (used for weather information)
- Reliability (indicator of the measurement installation)
- Manipulation (used intern to indicate a change of the value)

For gathering the data from different sources the following methods are used:

- Query to an online database
- Query to a fixed connected database
- Reading from a local file
- Reading from a remote file

Even though these methods are defined in the abstract class the mapping of the received data to the data structure is defined locally in each instance, since this is strongly installation dependent.

2.3.4 FrOnT - Vehicle entity

To allow the whole variety of simulation tools to work in FrOnT, the traffic demand is stored vehicle wise. Thus, FrOnT stores vehicles with its position, speed, acceleration and a set of vehicle dynamics, such as maximum acceleration, desired acceleration, maximum deceleration and desired deceleration.
Next to the vehicles state (position, speed) and its dynamics (trucks, cars, vans), a vehicle is described as well by its driver. Instead of including the behavioral aspects

to the vehicle, an abstract driver class is added. This allows testing new driving behavior or traveling behavior models in the same surroundings without changing the vehicle properties. Another abstract class is attached to the vehicle to simulate the influence of car equipment such as routing systems, cruise control and others. The driver class and equipment class are further described in Chapters 4.4 and 4.3. Public transport (busses, trams) has not been included yet.

2.3.5 FrOnT - Node entity

The attributes of a node are in the first place the coordinates in x, y and z, to describe the network geometry. For handling internally each node has an identification number and for the user friendliness a node is also equipped with a name. Next to the location, a node is stored with a certain type. These types are:

- Origin (node is added to the OD-table and linked to a demand file)
- Destination (node is added to the OD-table)
- Both (if the node is an origin and a destination)
- Intersection (if the node is located at an intersection)
- Intermediate (if the node has no additional functionality)

If a node is of the intersection type, an additional name is given to the node. This name is used to link the intersection to a controller and to a set of rules which defines unchangeable traffic rules (e.g. priority rules) for the links connected to the intersection node.

2.3.6 FrOnT - Link entity

The definition of a link varies between different simulation approaches. Hoogendoorn et. al. (2002) developed a data structure for links that could be used by different models. The FrOnT – Link class is based on this structure. The stored attributes of a link are:

- ID (internal number)
- Name (for an easier identification for the user)
- Description (descriptive information)
- NodeFrom (starting node of the link)
- NodeTo (end node of the link)
- Length (length of the link in meters)

- Shape (straight, curve)
- Radius (radius of the curve)
- Nr_lanes (amount of lanes)
- Nr_cells (defines the number of cells if using a cell transition model)
- Cell_length (if the cell length is given, the amount of cells is computed automatically)
- Lane_Array (array of lanes)
  - Capacity (array for user class specific capacity values used in macroscopic and mesoscopic models)
  - Width
  - Max_Speed
  - Lane_closure (ability of closing lanes for specific or all traffic)
- Detector_Array (array of detectors)
  - Name
  - Speed
  - Volume
  - Height
  - Vehicle class

This detailed description of the links of the network can feed different kinds of simulation models from a microscopic approach to a macroscopic approach and guarantees the flexibility of FrOnT.

2.3.7 FrOnT - Network entity

At the top of the class diagram is the FrOnT – Network class. Basically, this class contains the whole network information in the form of the above described classes. Additionally, the class contains a name and description for easier identification. The reason for including this class without much further addition is the possibility to handle of a network in FrOnT uniformly. So if changes occur in the data structure or additional classes are added, the handling in FrOnT is not changing, since the common interface from FrOnT – Network is used. This is a basic concept of object oriented programming and allows working with generalized objects where underlying changes can be done without further changes in the other software parts.

2.3.8 Data storage in FrOnT

Data storage in FrOnT is done within a relational database, which is a database structured in accordance with the relational model, introduced by Edgar Codd (1970). We chose a relational database over an object database to allow easier access through non object oriented software as Matlab®.

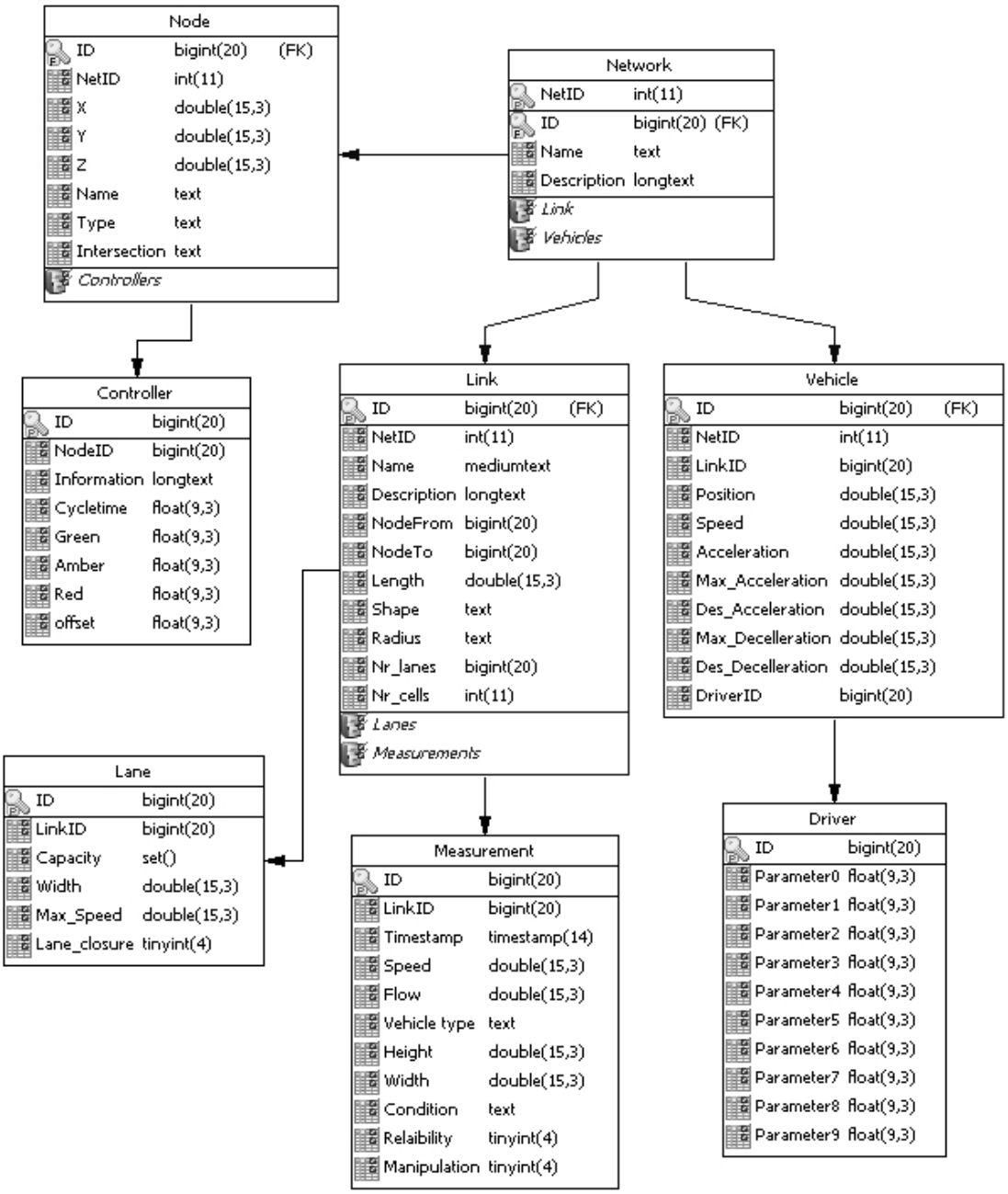

Figure 5: Entity relationship diagram of the FrOnT database

The internal FrOnT data structure is transformed into an entity relationship diagram (see Figure 5), which is a data model for high-level descriptions of conceptual data models, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams. Such models are typically used in the first stage of information-system design; they are used, for example, to describe information needs and/or the type of information that is to be stored in the database during the requirements analysis (Chen, 1976). The handling of the data is done by a relational database management system (RDBMS). FrOnT is using the MySQL system (www.mysql.com), published under the GPL license and therewith free for usage in the research field.
Now with the developed internal data structure for FrOnT and the database for storage, the next step is the data handling.

2.4 Data handling by Multi-agent Society

2.4.1 Basic concept

The basic concept for the data handling in FrOnT is that collector agents, distributed over the network, collect real time measurement information from the roadside systems, databases and weather stations and pass this data to a control agent .

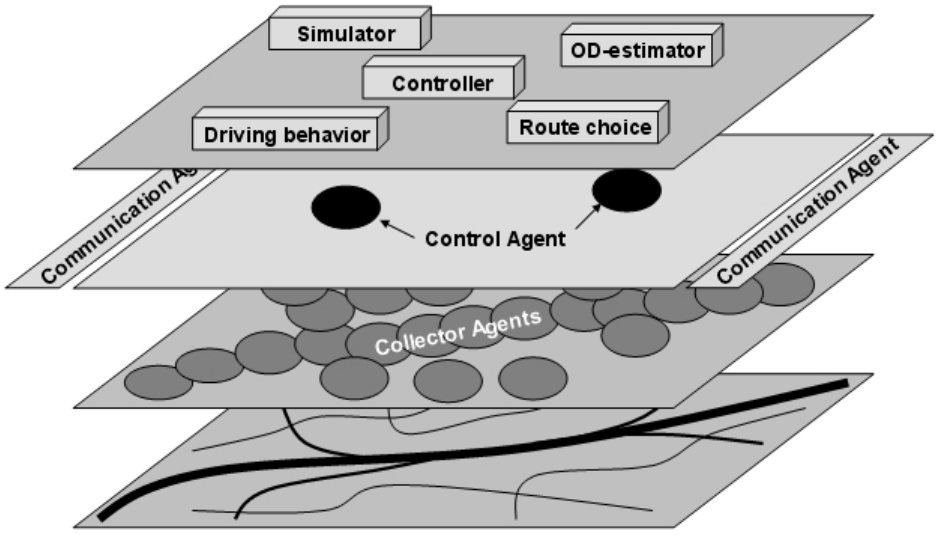

Figure 6: Agent society used in FrOnT

The control agent gathers all incoming information from the collector agents and is then able to feed the attached DTM tools with the required online information. For scalability reasons communication agents are introduced to allow several units of one control agent and its collector agents to communicate with each other. In the following the agents get defined in more detail.

2.4.2 Collector agent

For each data source, like traffic measurements, weather data or incident data, the multi agent systems creates a new Collector agent. The attributes for this agent are set by the network editor (described in 5.2) and include the location of the data source, the frequency of data updates, reasonable boundaries for the data and the protocol to receive the required information. When the agent gets started, an internal timer triggers the data request to the data source, which is mostly in form of a database query.
If the data arrives in a predefined time limit the collector agent pre checks the data. If the data is outside the defined reasonable boundaries it gets marked as unreliable. This method filters out outliers in the measurements which can occur and would severely change the simulation process. If the data lies close to the boundaries the data gets an incident flag which indicates that either the measurement is unreliable or that the change in traffic conditions signalizes an incident in the network. If no data arrives in the time limit, an empty dataset with an unreliable flag is stored. In any other case the data is stored as it is. The data of the last 5 requests is stored in the collector agent.
After the data is pre processed the collector agent looks up the control agent of its unit and transmits the data.

2.4.3 Control agent

The control agent as the central point of each unit collects the data from all collector agents and triggers the simulation by given protocols. The usage of protocols allows to use applications running on different systems (Windows, Mac OS, Solaris, Linux) and written in different languages (Java, C++, C#, Matlab, and others).
Even though the simulation process is independent from the multi agent framework, the control agent determines which tools are used. So if no specific algorithm is defined in the framework, the control agent can lookup all simulation tools which are available and chose one. The look-up of simulation tools is done via “yellow pages”, a listing of available modules, sorted by the task they can perform. Since this procedure is dependent on the implementation platform we will pick up this topic in 2.6. This

allows the easy exchange of all components of the system. The same mechanism is used for the robustness of the system. If a module is not responding anymore it will be replaced by an alternative tool in the network. In the case that no alternative can be found the system halts and informs the user. To enable even more robustness and a smooth exchange, the control agent buffers the simulation states in so called key frames. A key frame is the simulated traffic situation which is taken as a snapshot in defined frequency. This allows a restart of the simulation from this point.

When different scenarios are calculated, the control agent distributes the simulation tasks over all reachable simulation modules. Therefore, the scalability of the system is no problem. Additional simulation modules which are started in the computer network can be looked up easily. Nevertheless, when for any reason the computation time exceeds a critical value, the control agent will limit the simulation tasks and inform the user.

2.4.4 Communication agent

For larger networks it can be useful to distribute the simulation over the computer network to lower the computation load. FrOnT allows this feature and communication agents ensure that the traffic situation in the overlapping areas of the partial networks is transferred. That means that a communication agent does the same as a collector agent, except that the data source is another simulation unit. It collects the actual status of the link and informs the attached unit. Like a collector agent, the communication agent has to ensure the consistency of time. So the slowest unit of the simulation determines the speed of all units. Due to slight oscillations in the computation time the communication agents can buffer up to 50 time steps. If the buffers are full, the communication agent sends a message to its control agent and the faster simulation is halted. If units are on hold longer than a pre defined threshold the user will be informed, so that the area of the units can be changed to balance the computation load.

To meet our predefined requirement of robustness of the system we introduce one more type of agent to the society, which task is purely backing up the communication, so that the system can recover from a malfunction of one of the components during runtime. These *recovery agents* mirror the communication in the system and buffer them, so that the system can be reset to an earlier stage. We describe the process further in 2.5

2.4.5 Working scheme of FrOnT

When transforming the above described ideas into a working scheme, it resolves in a four layer model shown in Figure 7.

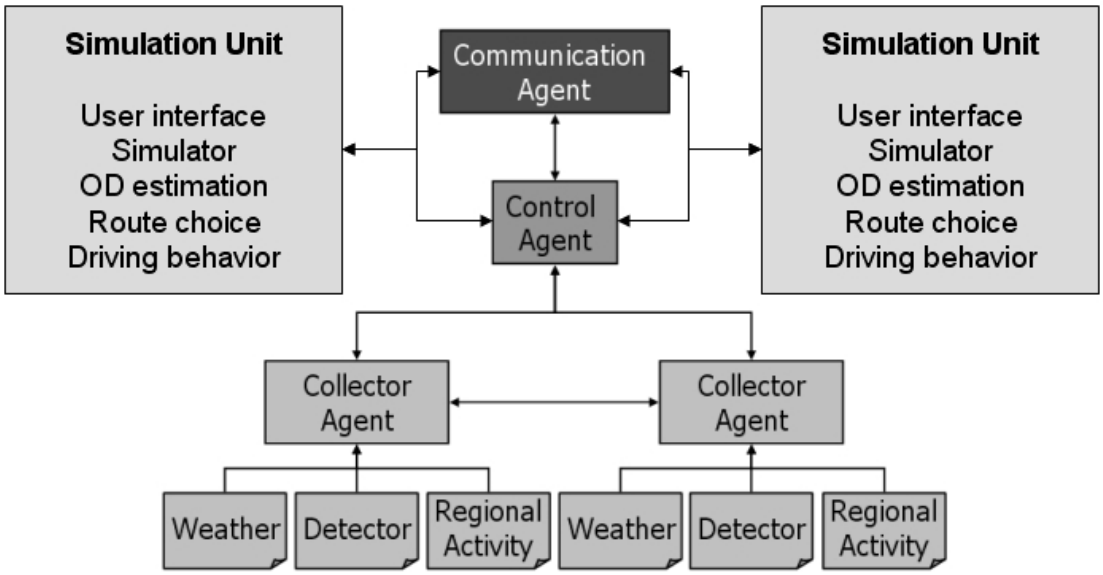

Figure 7: Structure of a FrOnT unit

The bottom layer consists of sensors that transmit the data to a data layer, in which the collector agents gather the information. The data is then sent to the control agent in the knowledge layer. Knowledge layer, because there the local measurements get combined and this layer will be the docking point for the simulation model. The top layer is for the communication. The control agent can feed one or more simulation units in the network and controls the simulation task. The communication agent's task is to communicate the overlapping network parts between the simulation units. This lowers the computation load for the control agent. Since communication is important for the whole system, its structure is described in the following. The recovery agents are not included in the unit structure since they just belong to an underlying backup system for the communication and will be described further in the following.

2.5 Communication structure

The communication between the agents is the most important thing to realize the quality of information the system should provide. There are three communication networks involved. First the horizontal communication lines which enables Control

Agents or Collector Agents to communicate on their own layer. For the communication between the layers a one directional channel is constructed, which enables the Collector Agents to provide the information to the Control Agents. A grid of Recovery Agents mirrors the communication of the whole system and buffers it. They are not connected to any other agents, but listen to the communication between them and store this information. Preferable these agents run on a different computer to avoid a total loss in case of a power failure. During the mirroring process (sending the data through the network) the network protocol (TCP/IP) ensures the error free transmission.

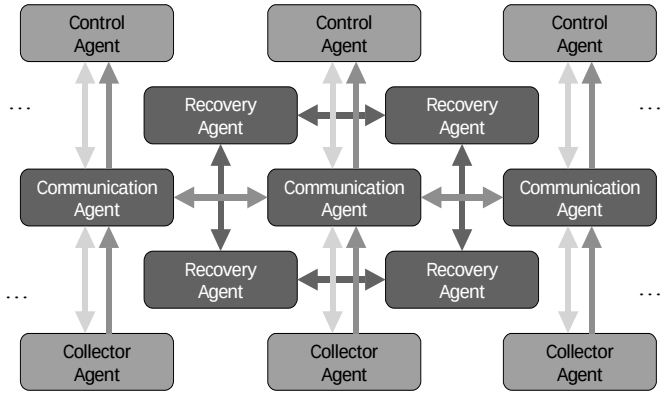

Figure 8: Communication structure in FrOnT

The external communication is based on a standardized data stream, which is provided by each Communication Agent. Besides, each Communication Agent can submit queries to others to get special data or to set the communication parameters. This way of coupling and building knowledge networks made it possible to distribute a big network on different computers.

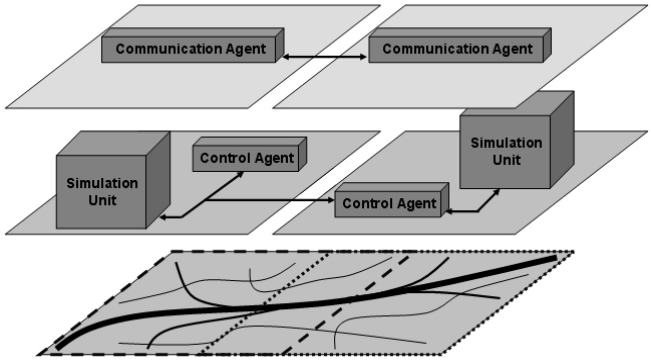

Figure 9: Coupled simulation units

After the design, the following will focus on the implementation decisions made for FrOnT.

2.6 Implementation

2.6.1 Decision of the system architecture

The goals of the software design determine the need for a distributed multi-agent system. Agents should be distributed in the network and remote method calls should enable the information flow as well as the start of processes in the simulation model. The scalability factor in the design leads to a peer to a peer (P2P) system which would offer most advantages. In contrary to the well known client-server (C/S) models, where the role between servers (resource providers) and clients (resource requester) is clearly distinguished, a peer to peer model mixes the roles. Each node can initiate the communication, be subject or object for a request, be proactive and provide capabilities. In such a system the agents and simulation modules have the ability to discover each other during runtime. Computers, running agents or simulation modules can enter, join or leave the network anywhere and any time. The drawback of the system being fully distributed across the network is that the complexity and bandwidth tends to grow exponentially with the number of connected computers. The absence of any reference of how many computers are connected and where they are in a pure P2P system makes it difficult to maintain the coherence of the network and to discover the offered functions in the network. Also security is quite demanding as each node is entitled to join the network without any control system.

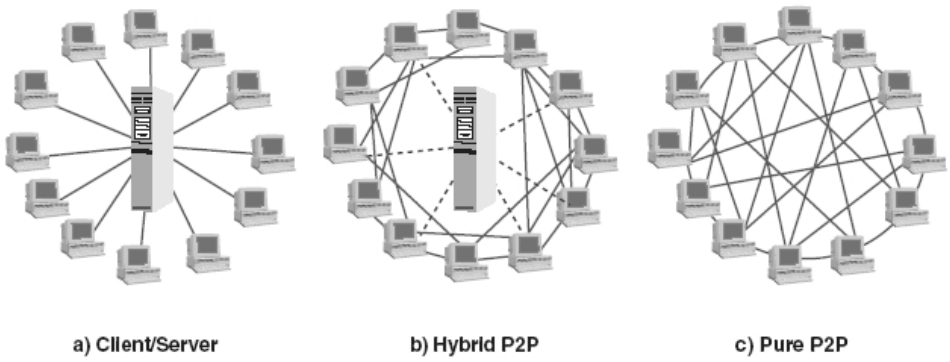

Figure 10: Comparison of different model structures

By choosing hybrid P2P architecture (see Figure 10) this drawback can be overcome. In this architecture one computer is added with special services, which provide a simplified look-up for specific computers or functions in the network, like the white pages for phone users, as well as the capabilities to find a computer in the network to perform a certain task, which would be the equivalent of the yellow pages. This creates less traffic and by adding a registration and authentication of joining computer it also increases the security of the whole system.

2.6.2 Programming language and runtime environment

After the design of the system and the architecture is chosen it is time to decide about the runtime environment and the programming language of the system. To follow the principle disparate of independency JAVA was chosen as the main programming language of the system. JAVA allows a platform-independent development and also assures the easy use of cross-platform programming if that is needed.
The main requirement of the proposed system is the distribution in the network and so a middleware is needed. Middleware is a general term for any programming that serves to "glue together" or mediate between two separate programs. A common application of middleware is to allow programs written for access to a particular database to access other databases (well know from Internet services).

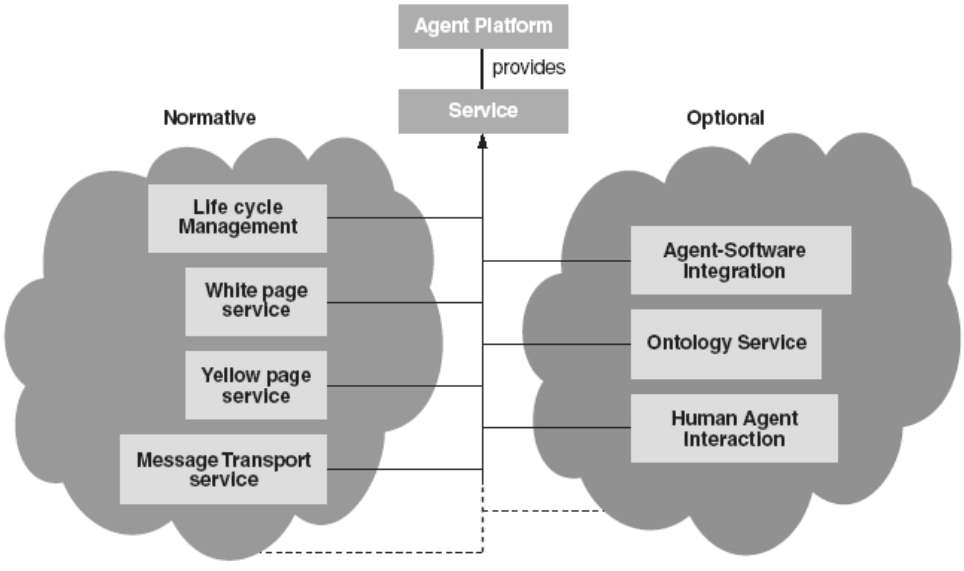

Figure 11: FIPA standard for services provided by a platform

Middleware programs provide messaging services so that different applications can communicate. The systematic tying together of disparate applications through the use of middleware, is known as enterprise application integration. Next to a variety of

commercial middleware, the amount of independent and GNU licensed (http://www.gnu.org) middleware is increasing. So to avoid a dependency to a commercial product non-commercial middleware is used.
The middleware Java Agent Development Framework (JADE), developed by the Telecom Italia Lab and published under the Lesser GNU Public License (LGPL), suits all the requirements stated above and has so been chosen for the development. JADE allows a fast development of distributed agent system by providing standard services for communication and life cycle management of the agents according to the standards of the Foundation for Intelligent Physical Agents (FIPA). The full standard can be found on the FIPA website: www.fipa.org. An overview of that standard is shown in the Figure 11.
The communication between the agents offered in JADE is done by the Agent Communication Language (ACL), also a standard from FIPA. It is based on the speech act theory (Searle, 1969) and on the assumptions and requirements of the agent paradigm. This paradigm is based on the agent abstraction, a software component that is autonomous, proactive and social:

- *autonomous*: agents have a degree of control of their own actions, they own their own thread of control and, under some circumstances, they are also able to take decisions
- *proactive*: agents do not only react in response to external events, for instance a remote method call, but they also exhibit a goal directed behavior and, where appropriate, are able to take initiative
- *social*: agents are able to, and need to, interact with other agents in order to accomplish their task and achieve the complete goal of the system.

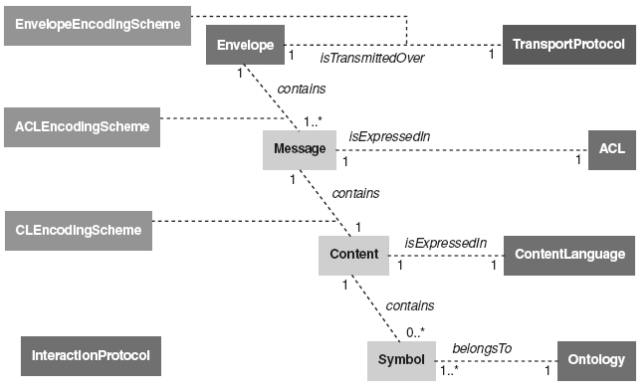

Figure 12: Communication model standard from FIPA

Common patterns of conversations define the so-called interaction protocols that provide agents with a library of patterns to achieve common tasks, such as delegating

an action, calling for a proposal and so on. The standardization of this protocol is shown in Figure 12.
So with the chosen programming language and middleware for the development the further section describes the implementation of the multi-agent system by using the JADE environment.

2.6.3 System description

We separate the description of the implementation in two parts. First we create the FrOnT runtime environment and in a second step we focus on the interaction between FrOnT and the simulation tools attached to the framework.

The FrOnT runtime environment consists of a FrOnT MAIN node and one or more other FrOnT nodes. The MAIN node is equipped with a Control Agent, a Communication Agent, the FrOnT database for the network information and the white pages and yellow pages services. These services register tools in the framework according to there name (white pages) or the functions they offer (yellow pages). With these services, every node in the framework can locate other nodes and tools they offer. The communication between the nodes is handled by the used middleware.

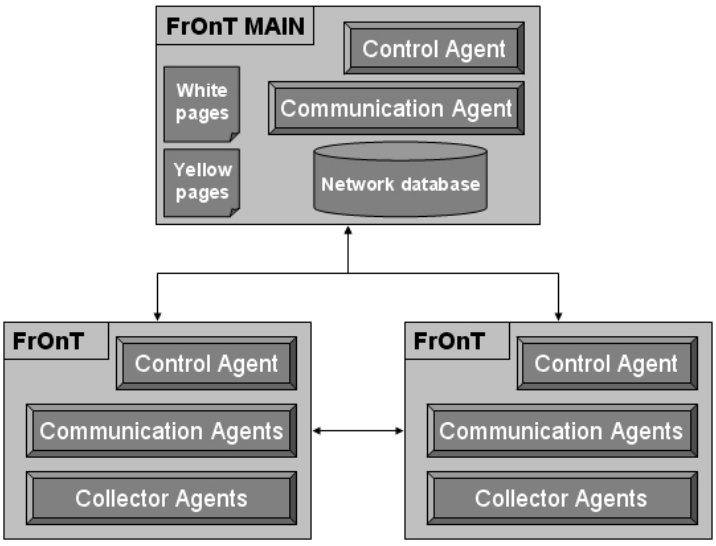

Figure 13: Connection between the MAIN node of FrOnT and other nodes

To this basic distributed system we are going to add the services for simulation. The simulation tools we kept in a unit are now being separated and added to the nodes as

service. These services will run on a node and being registered at the MAIN node for usage. The registering process is done by the middleware automatically. There are five different services:

- User interface (for network editing and viewing)
- Cellular automata for traffic simulation
- OD estimation
- Route choice
- Driving behavior

These services together form the microscopic online simulator MiOS to be described in Chapter 4. They interchange information via the FrOnT runtime environment and need to be started and register before a simulation. Due to the design of the framework it is not necessary to start all services on a local computer. Furthermore, each service can run on an independent computer in the network to distribute the computational load, which varies significantly between the services. While the user interface is idling most of the time, the computational load of the other services increases with the amount of individual cars that have to be handled. The startup procedure is illustrated in Figure 14.

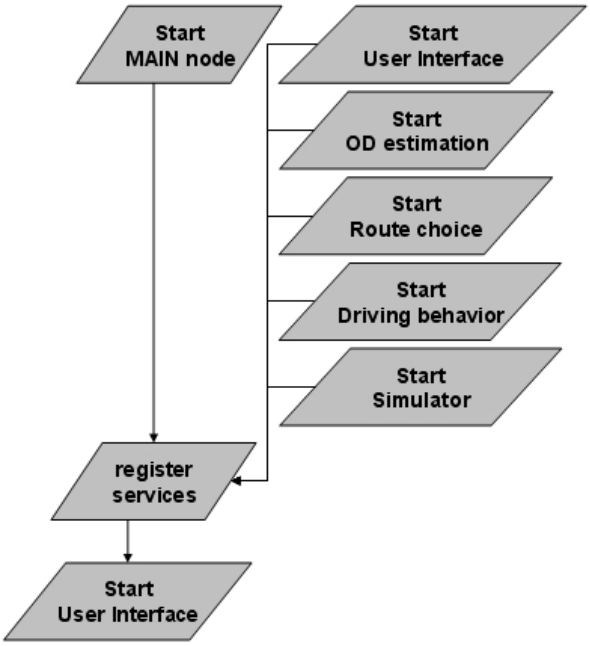

Figure 14: Startup procedure of FrOnT on a single machine

If further nodes are started the procedure will not start the user interface, but instead register at the MAIN node of FrOnT. After the startup procedure the user can define networks and start simulations. Simulations are done by combining all service in the network. Therefore, it is essential to implement a good communication structure (see Figure 15).

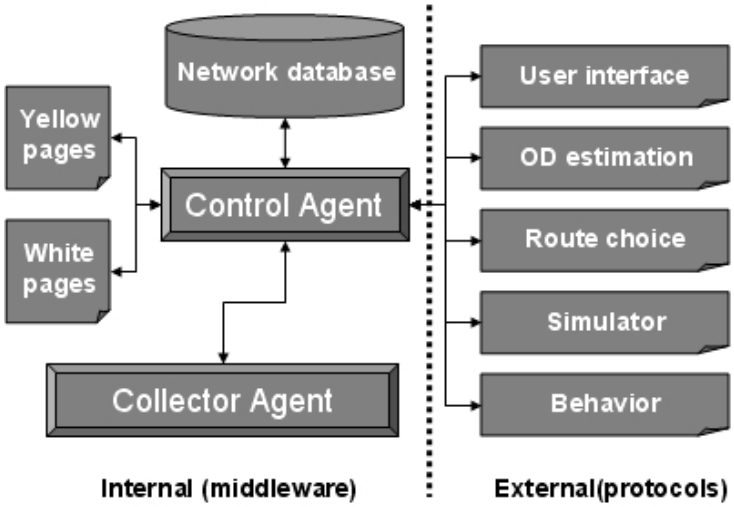

Figure 15: Communication structure for a simulation process in FrOnT

The communication protocols (described in Appendix B) support any kind of LAN (local area network) connection. A inter process communication as used with multi-processor machines is not included in the design, because the costs for these systems seem inappropriate for the task. A computer cluster, consisting of several computers however, allows an easy and considerably cheap possibility to scale the system to every needed level.

In the following we show the data flow through the system in more detail for a better understanding of the working scheme. As shown in Figure 16 all components needed for a simulation task are individual modules. With this design it is possible to exchange single or multiple parts to test new algorithms in a stable environment and can serve as a platform for performance tests of existing algorithms. This feature also allows keeping the system up to date with the state of research by simply exchanging the modules and not changing to a different system. That means in practical terms that newer and better algorithms will allow better network operation without new training of the staff, using the system.

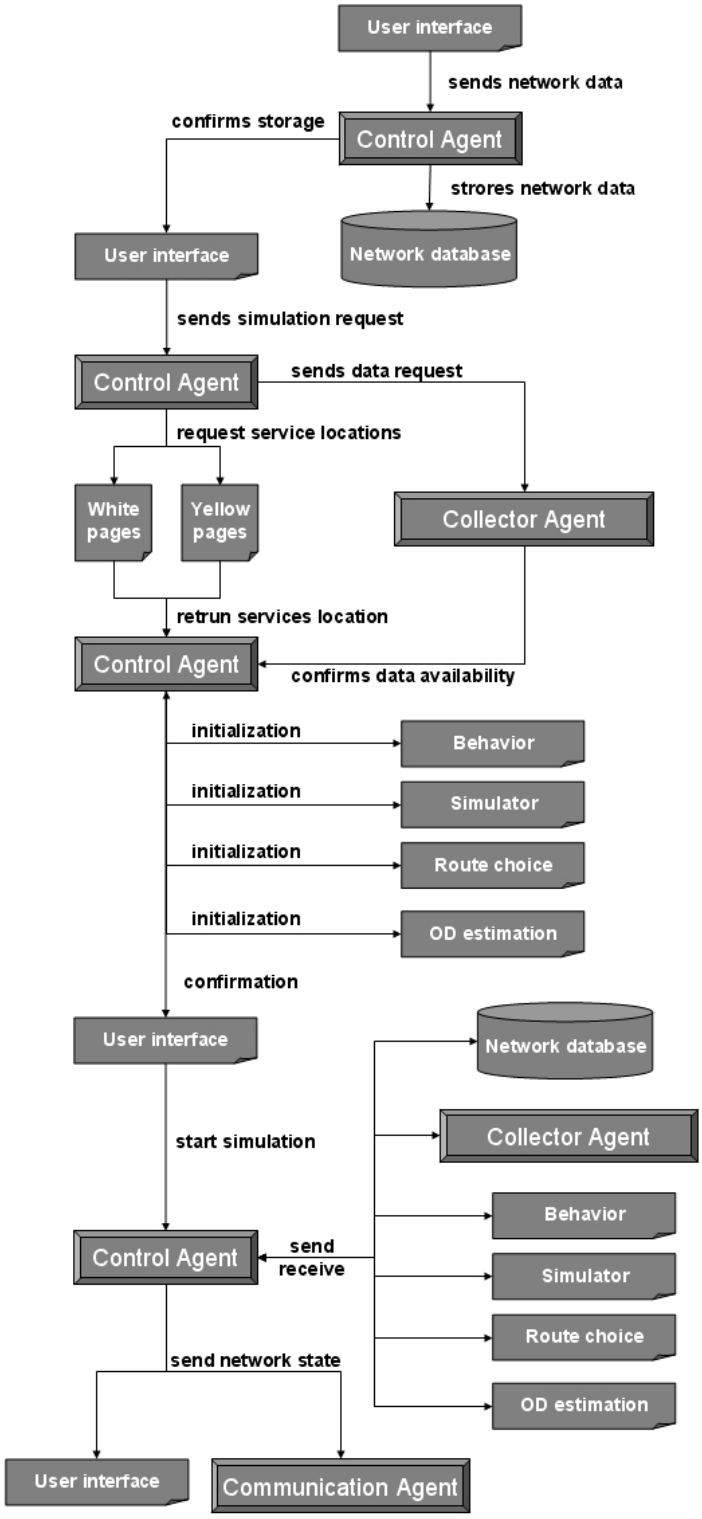

Figure 16: Dataflow for starting a simulation in FrOnT

2.7 Summary

Chapter 2 introduced the framework for online traffic management (FrOnT). It started with the definition of the internal data structure based on abstraction of the real world and describes the steps of object oriented modeling and database design. Afterwards, the multi agent society for data gathering and handling was introduced to enable FrOnT to collect data, transfer data between FrOnT systems and to process data with attached applications or algorithms in the computer network. With the finished design of the system, the Chapter dealt with the decision making process concerning the programming environment and its system architecture.
With the framework ready, the following Chapter will give an overview of existing real-time simulation models to identify the basic needs of an online simulator running in the FrOnT environment.

3 Real time Simulation models

3.1 Real-time simulation in road traffic

After a brief introduction of dynamic traffic management and the design of a data storing and handling framework for network operation tasks in Chapter 2 this thesis continues with the design of microscopic online simulator for traffic forecasting to be used in the framework. Real time simulation for road traffic aims to reproduce the actual traffic state from measurements along the roadside. To determine the actual state of the traffic network, roadside measurements are used as input in simulation models, while data mining tools or pattern matching methods are used to fill the gaps of knowledge about the network flows. To identify the routes of the measured vehicle flows to the network it needs OD estimation methods or split fraction estimation. Together with a route choice model the simulation can then give an insight of a probable actual traffic state of the network.
In a further step, including the prediction of future traffic flows and OD relations, simulation can be used to predict the probable future traffic state and allows therewith to test different ITS measures and their effect on the network before applying them. That helps traffic control centers to guide the traffic in a more efficient way through the network. To be operational, such a model must be able to simulate the traffic multiple times faster than real-time to ensure a reliable support for the traffic control task. To determine the required modules to perform this task, available models on the market and development have been assessed and are described in the following.

3.2 Available models on the market and in development

This section introduces the most important existing commercial real-time models and such under development. The information about the simulation programs has been verified by the authors or employers, as far as they reacted on a request to do so[1]. Among these real-time models there are four different approaches:

- microscopic models which simulate individual vehicles
- mesoscopic models which simulate platoons or distributions of vehicles

[1] Programs with an * behind the name are verified.

- macroscopic models, which simulates link flows
- agent based models, which use agents to represent single drivers

Table 1 gives an overview of the examined models and shows that researchers and companies put effort in developing such models to support network operations.

Model name	Traffic model	OD-estimation	Validation	References
AIMSUN	micro	External	Takapuna, New Zealand	Barcelo & Ferrer, 1999
BOSS/Metanet	macro	Estimated turning fraction	Amsterdam region, Netherlands	Messmer & Papageorgiou, 1990 Hoogendoorn, 2003
DynaMIT	meso	Kalman filtering based on historical OD and measurements	University of Virgina, USA	Ben-Akiva & Bierlaire, 2002
DYNASMART	meso	Least square using all counted flows	I-95 corridor, USA, Irvine	Jayakrishnan & Mahmassani, 1994; Mahmassani et al, 2004
MITSIM	micro	External	Amsterdam, Netherlands	Yang & Koutsopoulos, 1997
OLSIM	micro	None	North Rhine Westphalia, Germany	Chrobok et al, 2002
Paramics	micro	External	Toronto, Canada	Quadstone Group, 2004
Visum Online	macro	Path flow estimator	VMZ Berlin, Germany	PTV, 2001

Table 1: Comparison of available real-time simulation models

While gathering information on the mentioned we checked the possibility of using one of these models to run in the newly designed framework FrOnT. However, it shows, that these models were not available in their source code to make needed

changes or that the software design of the packages would have not allowed the distribution of the system in a way this thesis proposes. Therefore, we decided to develop and implement the necessary modules from scratch, with due to limitations in time, did not lead to a high end microscopic simulator at the state of the art, but allowed to follow some new ideas which lead to a basic set of simulation tools operating in the FrOnT framework, which each needs further attention for development.
In the following of this chapter the assessed models are described in more detail before Chapter 4 deals with the development of the microscopic online simulator MiOS with all its modules.

3.2.1 AIMSUN*

Package developer
The Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks (AIMSUN NG) is developed by TSS – Transport Simulation Systems, Barcelona. It is an offline simulator with an extension for real-time data input. AIMSUN NG follows a microscopic simulation approach that covers urban networks, freeways, highways, ring roads, arterials and any combination of them.

Model description
AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks), (Barcélo, Ferrer, 1999), is a microscopic traffic simulator that can deal with different traffic networks: urban networks, freeways, highways, ring roads, arterial and any combination of them. AIMSUN2 simulates traffic flows either based on input traffic flows and turning proportions, or on O-D matrices and route selection models. In the former, vehicles are distributed stochastically around the network, whereas in the latter vehicles are assigned to specific routes from the start of their journey to their destination. Different types of traffic control can be modeled in AIMSUN2: traffic signals, junctions without traffic signals (give way or stop signs) and ramp metering. Vehicle behavior models (car following, lane change, gap acceptance, etc.) evolved from the seminal Gipps models (Gipps, 1981 & 1986), are a function of several parameters that allow modeling of different types of vehicles: cars, buses, trucks, etc. They can be classified into groups, and reserved lanes for given groups can also be taken into account.
AIMSUN is embedded in GETRAM Generic Environment for Traffic Analysis and Modeling) (Barcélo, Ferrer, 1999) a simulation environment that has a user-friendly interactive interface through which the user can build the model using a traffic network graphic editor (TEDI), define the simulation experiments, and get an

animated view of the simulation run. Depending on the type of simulation new vehicles are input into the network according to flow generation procedures (headway distributions) at input sections, or using time sliced O-D matrices and explicit route selection. The simulation process includes in this case an initial computation of routes going from every section to every destination according to link cost criteria specified by the user. A shortest route component calculates periodically the new shortest routes according to the new travel times provided by the simulator, and a route selection model assigns the vehicles to these routes during the current time interval. Vehicles hold the assigned route from origins to destinations at least they had been identified as "guided" at generation time, then they can dynamically change the route en route as required for simulating vehicle guidance and vehicle information systems.

Validation
A validation for AIMSUN has been done at the Taharoto Road area, located in Takapuna, North Shore City, New Zealand. It was found that the calibration of the model was much more time consuming than expected, but that the validation results, based on offline data, were satisfying (Haas, 2001).

At the Access Management Conference in 2004, Jones Sullivan and Anderson did an assessment study on different models. They came to the conclusion that AIMSUN delivers good results in low and moderate traffic conditions, but needs a long calibration process for congested traffic situations. AIMSUN's car-following logic has been shown to be realistic in tests during the SMARTEST study (Algers, et al., 1997)

This information is validated by Pablo Barcelo, Marketing Manager at the TSS-Transport Simulation Systems, Spain.

3.2.2 BOSS/Metanet*

Package developer
A Decision Support System (DSS) called BOSS has been developed by several groups for the AVV Transport Research Center of the Dutch Ministry of Transport, Public Works and Water Management. The objective of BOSS is to provide the operators with conditional predictions on the future state of the traffic network under their supervision, given the current state of the network and conditional on the candidate control scenarios. BOSS itself is no simulation tool and uses Metanet as an integrated simulator.

Model description
BOSS uses real-time data to evaluate different scenarios of traffic management measures. Therefore it uses METANET (Messmer, Papageorgiou, 1990) as a simulator. BOSS is not working with OD matrices, but with turning fractions. It was found that in many cases the needed data for estimating turn fraction is not available. Recently, a method has been added to the model to estimate the missing measurement values, which allows then the determination of split fractions in the network (Hoogendoorn, 2005).

Validation
A validation of the BOSS system was done in the Netherlands (AVG et al, 1999) and Metanet has been validated in the area of Amsterdam (RAND Europe, 2001).

3.2.3 DynaMIT-R*

Package developer
DynaMIT (Ben-Akiva, Bierlaire, 2002) is a real time computer system for traffic estimation, prediction, and generation of traveler information and route guidance. DynaMIT is developed at the Massachusetts Institute of Technology. Its development is sponsored by the FHWA's Dynamic Traffic Assignment (DTA) project.

Model description
The key to DynaMIT's functionality is its detailed network representation (lane based instead of link based), coupled with models of traveler behavior. Through an integration of historical databases with real-time inputs from field installations (surveillance data and control logic of traffic signals, ramp meters, and toll booths), DynaMIT achieves:

- Real-time estimation of network conditions;
- Rolling horizon predictions of network conditions in response to alternative traffic control measures and information dissemination strategies;
- Generation of traffic information and route guidance to steer drivers towards optimal decisions

The state estimation module provides estimates of the current state of the network in terms of OD flows, link flows, queues, speeds and densities. This step represents an important function of DTA systems, since information obtained from the traffic

sensors can vary depending on the type of surveillance system employed. The main models used by the State Estimation module are:

- A demand simulator that combines real-time OD estimation with user behavior models for route and departure time choice.
- A network state estimator (also known as the supply simulator) that simulates driver decisions and collects information about the resulting traffic conditions.

The demand and supply simulators interact with each other in order to provide demand and network state estimates that are congruent and utilize the most recent information available from the surveillance system. Demand estimation in DynaMIT is sensitive to the guidance generated and information provided to the users, and is accomplished through an explicit simulation of pre-trip departure time, mode and route choice decisions that ultimately produce the OD flows used by the OD estimation model.
One of the inputs to the OD estimation model is a set of assignment matrices. These matrices map the OD flows from current and past intervals to link flows in the current interval. The assignment fractions therefore depend on the time interval, and also on the route choice decisions made by individual drivers. The flows measured on the network are a result of the interaction between the demand and supply components. If necessary DynaMIT iterates between the network state estimation and the OD estimation models until convergence is achieved. The output of this process is an estimate of the actual traffic conditions on the network, and information about origin destination flows, link flows, queues, speeds and densities.
The traffic information and guidance generation function uses the predicted traffic conditions to generate information and guidance according to the various ATIS in place. Traffic control is loosely coupled with DynaMIT in the current version of the system. Control strategies are assumed to be generated outside the DTA system, using the predictions as an input. (Balakrishna, 2006)

Validation
DynaMIT has been evaluated in several places across the United States. Balakrishna (2006) discuss the following:

- Irvine, CA,
- Los Angeles, CA,
- Hampton Roads, VA (Park et al, 2006)
- Lower Westchester County, NY

Next to its evaluation, DynaMIT has been applied in the following places:

- Southampton, UK
- Vienna, Austria
- Intercity network of NorthWest Switzerland,
- Boston, MA

3.2.4 MITSIMLab

Package developer
MITSIMLab is a simulation-based laboratory that was developed for evaluating the impacts of alternative traffic management system designs at the operational level and assisting in subsequent refinement. MITSIMLab was developed at the MIT Intelligent Transportation Systems (ITS) Program. Professor Ben-Akiva, Director of the ITS Program at MIT, and Dr. Koutsopoulos, from the Volpe Center, were co-principal investigators in the development of MITSIMLab. Dr. Qi Yang, of MIT and Caliper Corporation, was the principal developer. Recently an open source version of MITSIMLab became available (http://ngsim.camsys.com).

Model description
MITSMLab is a synthesis of a number of different models and has the following characteristics: represents a wide range of traffic management system designs; models the response of drivers to real-time traffic information and controls; and, incorporates the dynamic interaction between the traffic management system and the drivers on the network.

The various components of MITSIMLab are organized in three modules:

- Microscopic Traffic Simulator (MITSIM)
- Traffic Management Simulator (TMS)
- Graphical User Interface (GUI)

A microscopic simulation approach, in which movements of individual vehicles are represented, is adopted for modeling traffic flow in the traffic flow simulator (MITSIM). This level of detail is necessary for an evaluation at the operational level. The Traffic Management Simulator (TMS) represents the candidate traffic control and routing logic under evaluation. The control and routing strategies generated by the traffic management module determine the status of the traffic control and route guidance devices. Drivers respond to the various traffic controls and guidance while interacting with each other.

The role of MITSIM is to represent the "world". The traffic and network elements are represented in detail in order to capture the sensitivity of traffic flows to the control and routing strategies. The main elements of MITSIM are:

- *Network Components:* The road network along with the traffic controls and surveillance devices are represented at the microscopic level. The road network consists of nodes, links, segments (links are divided into segments with uniform geometric characteristics), and lanes.

- *Travel Demand and Route Choice:* The traffic simulator accepts as input given time-dependent origin to destination trip tables. These OD tables represent either expected conditions or are defined as part of a scenario for evaluation. A probabilistic route choice model is used to capture drivers' route choice decisions.

- *Driving Behavior:* The origin/destination flows are translated into individual vehicles wishing to enter the network at a specific time. Behavior parameters (such as desired speed, aggressiveness, etc.) and vehicle characteristics are assigned to each vehicle/driver combination. MITSIM moves vehicles according to car-following and lane-changing models. The car-following model captures the response of a driver to conditions ahead as a function of relative speed, headway and other traffic measures. The lane changing model distinguishes between mandatory and discretionary lane changes. Merging, drivers' responses to traffic signals, speed limits, incidents, and toll booths are also captured. Rigorous econometric methods have been developed for the calibration of the various parameters and driving behavior models.

Validation

MITSIM was tested in SIMLAB using several networks, including a 6-mile stretch of I-880 around Hayward, California and the system was demonstrated in a case study using the A10 beltway in Amsterdam. A detailed validation report of MITSIM can be found in Yang and Koutsopoulos (1997)

MITSIMLab is currently used in applications in the city of Stockholm, Sweden, funded by the City of Stockholm Real Estate and Traffic Administration, which is

responsible for traffic planning and operations within the city. Initially, MITSIMLab was evaluated for its applicability in that city. As part of the project, MIT enhanced the simulation models and calibrated the model parameters to match the observed conditions in Stockholm. Validation of the simulation model was performed by the Royal Institute of Technology (KTH) in Stockholm.

3.2.5 DYNASMART-X

Package developer

DYNASMART-X (Jayakrishnan, Mahmassani, 1994) is a real-time Traffic Estimation and Prediction System (TrEPS) for effective support of Advance Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS). DYNASMART-X is developed at the University of Texas at Austin and Maryland and sponsored by the FHWA's Dynamic Traffic Assignment (DTA) project.

Model description

DYNASMART-X interacts continuously with multiple sources of real-time information, such as loop detectors, roadside sensors, and vehicle probes, which it integrates with its own model based representation of the network traffic condition. The system combines advanced network algorithms and models of trip-maker behavior in response to information in an assignment-simulation-based framework to provide:

- Reliable estimates of network conditions;
- Predictions of network flow patterns and travel times over the near and medium terms in response to various contemplated traffic control measures and information dissemination strategies;
- Routing information to guide trip makers in their travel

Consistency checking and updating is an important function incorporated in DYNASMART-X to ensure consistency of the simulation assignment model results with actual observations, and to update the estimated state of the system accordingly. Another external support function is intended to perform the estimation and prediction of the origin destination (OD) trip desires that form the load onto the traffic network and are, as such, an essential input to the simulation-assignment core. External information about the traffic network is made available on a regular, periodic basis.

The DYNASMART-X system operates on these data and provides ATMS/ATIS normative outputs in response, also on a periodic schedule. DYNASMART-X, however, does not poll sensors and does not directly update controller timing; nor does it provide route guidance to individual travelers. Instead, it interfaces to a set of external real-time functions that perform these tasks. Under normal operation, a set sequence of real-time software modules must run each time that DYNASMART-X produces an output.

Exception handling is therefore a major consideration for DYNASMART-X. When external events require recalibration of the model parameters to achieve greater consistency between internal values and externally observed quantities, deadlines must still be met, and the system may enter a degraded mode of operation where the ATMS/ATIS outputs can continue to be provided, but with a reduced degree of system wide optimality. This type of strategy is common in complex real-time systems.

Validation

A detailed testing of DYNASMART-X is presented in Mahmassani et al (2004). This report describes the application of DYNASMART-X, to the Irvine test bed network and associated traffic management center (TMC) surveillance data. The basis for the calibration and evaluation is the actual traffic data provided by the University of California-Irvine in Orange County, California. The calibration and evaluation objectives are addressed through off-line testing activities. Off-line refers to the use of actual real-time data without “live” interfacing with the actual TMC. Instead, the testing is conducted *as if* the system was interfacing in real-time.

In collaboration with the Maryland State Highway Administration, DYNASMART-X has been used to estimate and predict the traffic network conditions and travel times along the I-95 corridor between the northern DC area (from the north Beltway) to the southern edge of Baltimore. In addition, a test is currently underway in Houston's TranStar TMC.

3.2.6 OLSIM

Package developer

OLSIM is a microscopic online simulation model that has been developed at the University Duisburg – Essen in Germany. It consists of a cellular automata model,

which is used to estimate the traffic state of the current traffic situation of the motorway network in North Rhine Westphalia, Germany.

Model description
The input for the model is generated by automatic data detection units on the motorway which are updated every minute. Vehicles are generated at the borders of the network and turning fractions at on- and off-ramps are used for the simulation (Chrobok, Pottmeier, Marinosson & Schreckenberg 2002). The methods of data cleaning and integration are not published. Since the cars are simulated without destination and just follow predefined turning fractions, the model has no OD estimation or prediction. Plans to extend the model with the functionality of predictions are made but so far nothing has been published.

Validation
The model is continuously running for the Autobahn network in North Rhine Westphalia and the results can be found on the internet. There is no report on validation of these results available.

3.2.7 Paramics

Package developer
There are two different versions of Paramics available. One version, called S-Paramics, is distributed by SIAS Limited while the other version is distributed by Quadstone. Since S-Paramics is not offering online simulations the following describes the Paramics microscopic simulation suit developed by Quadstone. It was designed as an offline simulation model for planning purposes, but through the extension of the *Programmer* module it allows to include online measurements to the simulation.

Model description
The core Paramics tool set consists of *Modeller*, *Processor*, and *Analyser*. *Modeller* provides network build, simulation, and visualization via a graphical user interface (GUI). Geographic and travel data is input to the program which then simulates the lane changing, gap acceptance and car following behavior for each vehicle and provides a statistical output capability that allows users to study the performance of their network and to obtain the information required to carry out standard transportation studies. *Analyser* reads output from the *Modeller* simulation and provides a GUI to select results for visualization and easy comparison with observed data. This visual interface to model statistics gives the user an output such as Level of

Service, queue lengths, turning and link flows etc. *Processor* sets up and runs the traffic simulation in batch mode without visualizing the network and vehicles through the GUI. This procedure is used once the model build is complete, to set sensitivity parameters and then collect sets of model results.

Programmer is a comprehensive development API for the Paramics suite. It is a tool available for users interested in microscopic traffic simulation. *Programmer* allows users to augment the core Paramics simulation with new functions. *Programmer* can be used for research, all aspects of ITS, real time connectivity and control, or customized model behaviors (Quadstone Group 2004).

A project performed at the University of California (Engberg et al, 2001) describes that the actual work with the programming interface of Paramics is very time consuming and finally they had to drop the intent to use Paramics and to delay the realization until a new release of Paramics.

Validation
Validations of Paramics have been performed with offline data in the following locations:

- Lower and Midtown Manhattan, Sydney CBD with SCATS signal control integration
- City models including Tampa, Miami, Toronto, Cologne, Paris, Orange
- Country and Greater LA
- Freeway networks including California, New Jersey, New York, Florida, Italy, France, South Africa, Germany
- Roundabout analysis including Hong Kong, New York State, Portland, Oregon, Adelaide Australia
- Public Transport Priority including Brisbane Australia, Beijing China,
- Toronto Canada

A full calibration and validation report on Paramics can be found in Oketch & Carrick (2005) Even though Paramics can be used for online simulation, no online application is reported so far, which emphasizes that extending an offline model with the possibility to input real-time measurement data is not enough to promote it as an online model.

3.2.8 VISUM Online / MONET

Package developer

PTV Germany developed an online version of their offline macroscopic traffic model VISUM, which was a part of the MONET project from Siemens. VISUM online (PTV, 2001) is a package of different software tools to keep the model adaptable to an existing infrastructure.

Model description

VISUM online includes multimode and multi-class assignment functionality known from standard transport models. Other models have been added to represent the dynamic nature of traffic. Besides pre-specified input data like coded road networks of various detail and surveyed Origin-Destination matrices, incoming online traffic data is the primary input. Information on traffic flows is gained from different sources like traffic actuated signal control units and dedicated traffic monitoring devices.

The current state of traffic flow is derived from the incoming traffic data being measured. Data completion in VISUM online is based on algorithms developed to estimate origin-destination matrices from traffic counts at cross sections. The Path Flow Estimator (Bell 1997) designed at the University of Newcastle is incorporated within VISUM online. An approximate (historical) O-D matrix and online traffic data (volumes and occupancy rates) are used to compute the most likely traffic situation.

Short term forecasts are generated for the next hour and long term forecast provide information for the next coming days. The first forecast is based on traffic flow simulation, the second one on dynamic assignment. Based on the current traffic conditions gained by data completion traffic states are forecasted every 15 minutes for the next one hour. The mesoscopic traffic flow simulation model DYNEMO is applied for short term forecasting of urban and rural road networks. Regarding movement of vehicles, DYNEMO is a mesoscopic model in the sense that the unit of traffic flow is the individual vehicle rather than the temporal and spatial aggregates used in static assignment models. Their movement is governed by the average traffic density on the link they traverse rather than the behavior of other driver vehicle combinations in the immediate neighborhood as in microscopic flow models.

For long term forecasts, volumes and volume dependent link travel times are computed and recorded separately for each of 96 quarter hour time slices. Route choice is governed by the expected trip time, summing over the expected future link travel times at the time when the vehicle will reach the particular link. Chosen routes and link travel times are adjusted simultaneously in an iterative procedure. The demand side in dynamic assignment is given in the form of hourly (or shorter) OD matrices rather than daily matrices typically used in strategic transportation planning.

Validation
The introduction of VISUM online for the traffic management during the EXPO 2000 in Hannover, Germany failed due to a lack of traffic information. A report on an application in Berlin can be found in (Vortisch, 2001). VISUM online is used in Munich for the planning and management of special events and recently got installed at the traffic control center in Utrecht, Netherlands. Evaluations of this application are not available yet.

3.2.9 Other developments

Other real-time models without further details should be mentioned due to completion of the overview:

- At the Fraunhofer Institute the online simulator CityTraffic (GMD AIS, 2000) has been developed. The core of the system is the combination of sensor-supported traffic monitoring systems which run in real-time, and a micro-simulation utilizing real traffic data based on a multi-agent approach.
- Within the framework of the DACCORD project a Statistical Traffic Model (STM) has been developed (Grol, 1997). The STM is a macroscopic traffic flow model that uses Kalman filtering techniques to track the current conditions in the network.
- DYNAMEQ (Mahut 2001; Mahut, Florian et al. 2002; Mahut, Florian et al. 2003) has been developed by INRO Consultants, who also developed the static equilibrium assignment model EMME/2.
- MATSIM (Cetin, Burri et al. 2003) (Raney, Cetin et al. 2003) has been developed by Kai Nagel et al. to simulate traffic dynamics of very large networks (such as all of Switzerland) much faster (200x) than real time.

Due to the fact that there is just limited published information on these models or the development has been stopped these models are not described in further detail.

3.3 Summary

Chapter 3 gave an overview of the basic concept of real-time simulation models and existing models on the market as well as models under development or former research projects. Overall it can be said that while offline models are well described in

literature, the online or real-time models are less present. This is maybe caused by the fact that the important online models are based on the former offline versions, which have been extended for online applications.

The amount of existing models reflects the need for support of online traffic management. The described models take different approaches and differ in possible network sizes and simulation speeds. The topic of OD estimation is taken out of half of the models and requires therefore an additional external tool to provide this data. Compared to the FrOnT framework of Chapter 2, the described models come in packages which are possible to extend, but under strong limitations. The new approach in FrOnT is the freedom to combine state of the art tools for all fields of online simulation to a most suitable system for the user. It overcomes the drawback of balancing out the pro and cons of program suites and allows the creation of an individual system according to the requirements set by the user.

Therefore, the commercial distribution of FrOnT would be different compared to the existing models. It is the environment to run modules from different suppliers and opens a new market for selling not only complete modeling packages, but single modules for various tasks of online simulation. This allows the specialization in a less wide area of research and could lead to overall better performing systems to deal with the growing traffic demand and to ensure an optimized traffic operation.

4 MiOS – A Microscopic Online Simulator

4.1 Introduction

To be able to use the FrOnT system for real-time simulation the microscopic online simulator MiOS has been developed. MiOS is a modular based simulation tool and consists of 6 modules:

- an online data interface,
- a traffic simulator,
- a driving behavior model,
- a route choice model,
- a OD estimation and prediction tool,
- and a postprocessor.

These modules together are able to simulate traffic networks fed by online measurement data and to forecast by simulation the probable future network state. Figure 17 gives an overview of the structure of the MiOS modules, which are independent in their operation and able to work together and exchange information via the FrOnT data handling and processing framework.

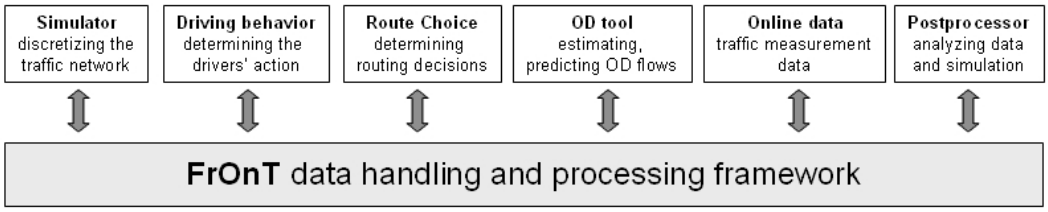

Figure 17: Structure of MiOS

In the following we describe the traffic simulator, the driving behavior model, the route choice model and the OD estimation and prediction tool in further detail before the data interface and post processing are described in detail in Chapter 0 in the form of a user manual of an application of MiOS, running in the FrOnT framework. The reader should keep in mind that the major topic of this thesis is the design of a modular based online simulator, running in a distributed network of data storage,

processing and handling. The following modules do not mean to extend the state of the art algorithms or to be fully validated. The development is based on curiosity for new methods and purely aims to develop modules to test the concept of the overall system architecture proposed in this thesis.

4.2 Traffic Simulator

To represent the vehicles' positions and dynamics in the network and to control their process a traffic simulator is used. Due to fast implementation and easy understanding a cellular automata was chosen, which is a well known mathematical concept and used in several fields of research. In this section we give a short overview of the history, the concept and the usage of CA models in traffic engineering. Based on that review we introduce the adapted CA model used in MiOS.

4.2.1 Concept of cellular automaton simulation

The mathematical concepts of cellular automata (CA) models can be traced back as far as 1948, when Johann Louis von Neumann introduced them to study (living) biological systems (Neumann, 1948). Central to von Neumann's work, was the notion of *self-reproduction* and theoretical machines (called *kinematons*) that could accomplish this. As his work progressed, von Neumann started to cooperate with Stanislaw Marcin Ulam, who introduced him to the concept of *cellular spaces*. These described the physical structure of a cellular automaton, i.e., a grid of cells which can be either 'on' or 'off' (Wolfram, 1983 & Delorme, 1998). The components of a cellular automaton are:

- *Physical environment,* to define the universe on which the CA is computed. This underlying structure consists of a discrete lattice of cells with a rectangular, hexagonal, or other topology. Typically, these cells are all equal in size; the lattice itself can be finite or infinite in size, and its dimensionality can be 1 (a linear string of cells called an elementary cellular automaton or ECA), 2 (a grid), or even higher dimensional. In most cases, a common—but often neglected—assumption, is that the CAs lattice is embedded in a Euclidean space (Maerivoet, 2005), in order to describe natural systems accurately on an ordinary scale.

- *Cells' states,* where typically an integer represents the number of distinct states a cell can be in, e.g., a binary state. Note that a cell's state is not restricted to such an integer domain (e.g., Z^2), as a continuous range of values is also possible (e.g., R^+), in which case we are dealing with coupled map lattices (CML) (Crutchfield et. al. 1987 & Kaneko, 1990). We call the states of all cells collectively a CAs global configuration. This convention asserts that states are local and refer to cells, while a configuration is global and refers to the whole lattice. In MiOS we use an object to define a cell's state. Therefore, we can use more than one parameter to describe the cell's state.

- *Cells' neighborhoods*, which locally determines the evolution of the cell. The size of neighborhood is the same for each cell in the lattice. In the simplest case, i.e., a one-dimensional lattice, the neighborhood consists of the cell itself plus its adjacent cells. In a two-dimensional rectangular lattice, there are several possibilities, e.g., with a radius of 1 there are, besides the cell itself, the four north, east, south, and west adjacent cells (von Neumann neighborhood), or the previous five cells as well as the four north–east, south–east, south–west, and north–west diagonal cells (Moore neighborhood). Note that as the dimensionality of the lattice increases, the number of direct neighbors of a cell increases exponentially (Maerivoet, 2005).

- *Local transition rule* (also called function) acting upon a cell and its direct neighborhood, such that the cell's state changes from one discrete time step to another (i.e., the system's iterations). The CA evolves in time and space as the rule is subsequently applied to all the cells in parallel. Typically, the same rule is used for all the cells (if the converse is true, then the term hybrid CA is used). When there are no stochastic components present in this rule, we call the model a deterministic CA, as opposed to a stochastic (also called probabilistic) CA. As the local transition rule is applied to all the cells in the CAs lattice, the global configuration of the CA changes. This is also called the CAs global map. Sometimes, the CAs evolution can be reversed by computing past states out of future states from deterministic CAs. By evolving the CA backwards in time in this manner, the CAs inverse global map is computed. If this is possible, the CA is called reversible, but if there are states for which no pre-cursive state exists, these states are called Garden of Eden states (Gutowitz et al., 1996) and the CA is said to be irreversible. Finally, when the local transition rule is applied

to all cells, its global map is computed. In the context of the theory of dynamical systems, this phenomenon of local simple interactions that lead to a global complex behavior (i.e., the spontaneous development of order in a system due to internal interactions), is termed self-organization or emergence in some cases. (Maerivoet, 2005)

4.2.2 Cellular automata of MiOS

When applying the cellular automaton analogy to vehicular road traffic flows, the physical environment of the system represents the road on which the vehicles are driving. In a classic single-lane setup for traffic cellular automata, this layout consists of a one-dimensional lattice that is composed of individual cells. Each cell can either be empty, or is occupied by *exactly* one vehicle. Because vehicles move from one cell to another, Traffic Cellular Automata (TCA) models are also called *particle–hopping models* (Nagel, 1996). An example of the tempo-spatial dynamics of such a system is depicted in Figure 18, where two consecutive vehicles *i* and *j* are driving on a one-dimensional lattice.

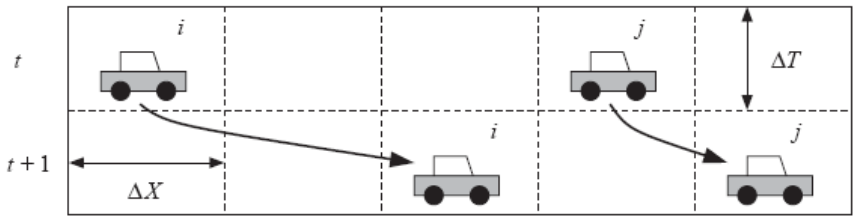

Figure 18: Schematic diagram of a traffic cellular automata (TCA) model

The typical discretization scheme of ΔT = 1 s and ΔX = 7.5m, corresponds to the average length a conventional vehicle occupies in a closely jam packed (and as such, its width is neglected), and a typical driver's reaction time (Nagel, 1992).
In MiOS we allow a vehicle to occupy several consecutive cells instantaneously for a better representation of the individual vehicles. This results in what is called a *multi-cell model* with a ΔT = 0.1 s and ΔX = 0.5m, corresponding to speed increments of $\Delta V = \Delta X/\Delta T$ =18 km/h. This means, that the representation of gaps, speed differences and vehicle types can be done more realistic than in the original model. Thus, MiOS is able to model different vehicle classes (see Figure 19) like cars (4.5 m or 9 cells), delivery vans (7.0 m or 14 cells), busses (12.0m or 24 cells) and trucks (17.0 m or 34 cells) (CROW, 2002). While the list of vehicles, differing in length, can be extended, the width of a vehicle is still not included, which means that there can only be one car

in the given lane width. That means that while changing lanes vehicles block two lanes during the maneuver.

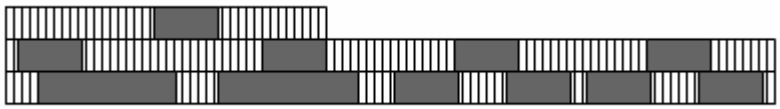

Figure 19: Illustration of a road stretch discretized by cells

This change however interferes with the computational performance of the cellular automata model following the classical approach. Since the update frequency in MiOS is ten times higher the computational load rises accordingly. However, the design of the framework the simulator runs in and taken into account the computational performance of a single PC nowadays the performance is still better than from older implementations in their times. The propagation of the individual vehicles in a traffic stream is described by means of a rule set that reflects the car-following and lane-changing behavior of a traffic cellular automaton evolving in time and space. The TCAs local transition rule actually comprises this set of rules. They are consecutively applied to all vehicles in parallel (called a *parallel update*). So in a classic setup, the system's state is changed through *synchronous position updates* of all the vehicles: for each vehicle, the new speed is computed, after which its position is updated according to this speed and a possible lane-change maneuver. Because time is discretized in units of ΔT seconds, an *implicit minimum reaction time* of ΔT is assumed in TCA models. With MiOS using 0.1s ΔT we have to compensate for the reaction time, which is about 1s. Thus, drivers get a reaction time between 0.7s and 1.4s after which they reassess their status.

For single-lane traffic, standard CA models assume that vehicles act as *anisotropic particles*, i.e., they only respond to frontal stimuli. So typically, the car-following part of a rule set only considers the direct frontal neighborhood of the vehicle to which the rules are applied. The radius of this neighborhood is usually taken large enough such that vehicles are able to drive collision-free. Typically, this radius is equal to the maximum speed a vehicle can achieve, expressed in cells per time step. From a microscopic point of view, the process of a vehicle following its predecessor is typically expressed using a *stimulus–response relation* (Maerivoet, 2005). This response is the speed or the acceleration of a vehicle; in TCA models, a vehicle's stimulus is mainly composed of its speed and the distance to its leader, with the response directly being a new (adjusted) speed of the vehicle. In a strict sense, this only leads to the avoidance of accidents. Some traffic flow models however, incorporate more detailed stimuli, such as anticipation terms (Toledo, 2003, Sukthankar, 1997). These forms of 'anticipation' only take leaders' reactions into

account, *without predicting* them. When these effects are taken into account together with a safety distance, strong accelerations and abrupt braking can be avoided. Hence, as the speed variance is decreased, this results in a more stable traffic stream (Eissfeld et. al., 2003 & Larraga et. al., 2004). But since recent work by Hoogendoorn (2005), Tampere (2004) and others show that drivers not only react on their direct predecessor. To set up update rules for the CA including recent research would lead to complex set of rules with deeply nested case distinction. Instead, MiOS uses a driving behavior model described in Chapter 4.3. The input for the behavior model is the surrounding of the vehicle determined by the CA model, covering a radius of twice the maximum speed a vehicle can achieve, expressed in cells per time step.

4.3 Driving behavior

Microscopic traffic flow models include mostly sub-models for driving behavior. Driving behavior models try to describe the driving task and are essential to predict the behavior of vehicle-driver combinations. The CA model in MiOS does not define this driving behavior and therefore, a model is needed to determine the drivers' action to perform the update in the simulator. Due to the independency of the driving behavior model from the simulator there are no restrictions according to the internals of the model except the inputs available and outputs needed for the CA model and we tried a new approach.
Thus, we presents first briefly the psychology of the driving task and developments in car following and lane changing models to give an overview of the state of the art in driving behavior modeling and the various approaches to model human actions before describing the driving behavior model in MiOS.

4.3.1 Psychology of driving behavior

The psychology of driving behavior is a complex topic and starts with the visual and audible perception. The driver perceives information from his or her surroundings and processes the information in three layers (see Figure 20).
The behavior of a driver can be distinguished in three levels, depending on the amount of conscious decision making. Nielsen et al (2003) makes the distinction in routine or skill based, rule based and knowledge based decision making. Skill based is decision making where the person is fully accustomed to a situation and is able to decide without conscious deliberation. Perceptions lead automatically to the right (routine) action.

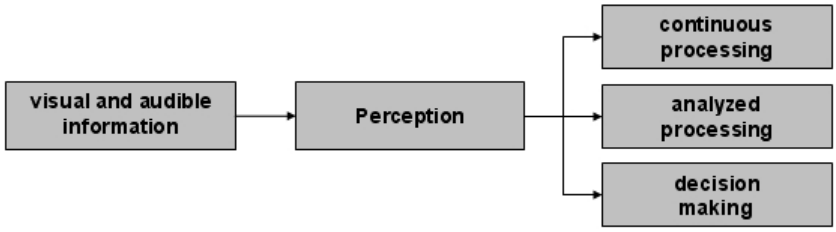

Figure 20: Perception of a driver

In rule based behavior people decide on certain standard knowledge that could be represented as rules: if I see x, than I decide y. Finally in knowledge based decision making people observe the situations, they do not recognize it directly and they think bout the most appropriate action to take. Skill based behavior is the quickest, knowledge based takes the most time. Most driver behavior is skill based.
For the visual perception, the driver is identifying the situation by a routine of eye movement. The more experienced a driver is, the eye movement gets more efficient and therefore lowers the needed mental capacity, since information from the periphery is filtered out. This leads to a better focus and allows the usage of more mental capacity to deal with unforeseen incidents along the trip (Jorgensen, 2002).
Continuous and analyzed processing effects car following, speed decision and route guidance. During the continuous processing the car following is determined by the visual angle of two fixed points at the rear of the leading car or the tail lights by night. The speed decision is based purely on the road geometry and traffic signs have no influence on this level. In terms of route guidance, the driver just follows known routes in this level of processing.

In the analyzed processing the driver evaluates the gain of lane changes and overtaking maneuvers and recognizes the restrictions of traffic signs along the road. This leads to a speed decision based on the leading vehicle as well as on the restrictions. Further the driver will react on mandatory route choice information.
In the decision making, the driver makes decisions about the departure time and mode choice depending on pre-trip information and plans the route, based on the knowledge of the network (Rothengatter, 1998).

Riemersma (1979) defines time horizons for these three levels of processing. The continuous processing of information is done with a preview in the order of maximally a few seconds, while the analyzed processing has a time span of some seconds to some minutes. Decision making is performed just occasionally and the horizon concerned may include the total duration of the trip.

Further he defines three main factors influencing the driver behavior:

- Situational factors
- Individual differences
- Other traffic

Situational factors can be divided in two categories: First the environmental factors like time of day, day of the week, weather and road conditions and second the individual factors like hurry, distraction, impairment, trip purpose, trip length and driving time. Other traffic can be divided in three different car-following zones defined between the free flow situation and congestion. There are also several individual differences of drivers like age, gender, risk-taking propensity and driving skills (Green, 2000) that influence the driving behavior.

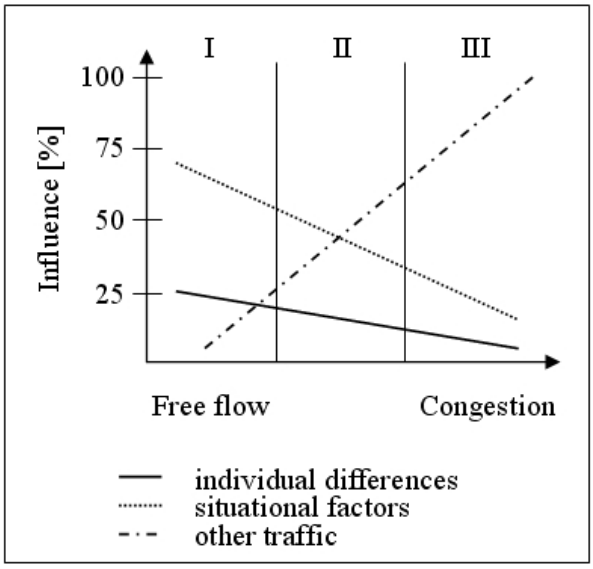

Figure 21: Influencing factors to driving behavior Riemersma (1979)

In congestion the situational and individual factors become more and more meaningless and the other traffic determines the behavior. In free flow conditions the situational and individual factors have the major part and the other traffic becomes meaningless, according to Riemersma (1979).
Van Winsum (2000) criticized, that engineers are tempted to add parameters to driving behavior models to fit them better to data sets without realizing the work on driving behavior in psychology and just little evidence of relations between different

drivers and there reactions. Boer (2000) claims that engineers ignore one or more of the following issues that characterize to observed driver behavior:

- Car following is only one of many tasks that drivers perform simultaneously and receives therefore only intermittent attention and control
- Drivers are satisfied with a range of conditions that extend beyond the boundaries imposed by perceptual and control limitation
- In each driving task drivers use a set of highly informative perceptual variables to guide decision making and control

Boer et al (1998) proposed a model in which the car following task is highlighted to elucidate these issues. In the ARCHISIM project (Espie, 1994) a driving behavior model was developed that does not follow a mathematical law, but is purely based on the interactions between road users (car drivers, pedestrians …). Their multi-agent approach (Doniec et. al., 2005) is under further development and shows the complexity of describing human behavior during the driving task. Next we are going to the mathematical models to describe driving behavior.

4.3.2 Car following

Car following models describe the process by which drivers follow each other in a traffic stream. In free flow conditions drivers do not influence each other. This situation can be described using knowledge on the speeds that drivers maintain when they are free. When densities increase, drivers have to adapt their driving to the other traffic. When a free driver approaches another vehicle and cannot overtake this vehicle, he will have to adjust his speed to the vehicle in front. This process of adaptation is determined by psychological abilities of the driver (what distance gap is perceived, what speed difference) and on his psychological state (aggressive driving, defensive, et cetera).
When a driver has fully adjusted to the vehicle in front, he will try to maintain a certain distance gap, which depends on his speed. The speed of the following vehicle (the follower) will be about the same as the speed of the vehicle in front (the leader). When the leader changes his speed, the follower will change his speed to regain a small speed difference and certain headway. This is modeled in a stimuli-response mechanism with the general form:

$$\text{Reaction}_{\text{follower}} = \text{sensitivity}_{\text{follower}} \cdot \text{stimulus} \quad (1)$$

The most well-known models are the Gazi-Herman-Rothery (GHR) model, collision avoidance models, the linear Helly model, action point models and more recently fuzzy logic based models. For a comprehensive overview of these models see Brackstone and McDonald (1997).
For calibration of these models studies by data collection with instrumented vehicles have been performed for example by Minderhoud et al (2002), McDonald et al (1994), Postans and Wilson (1983), Dijker et al (1997), Mardsen et al (2001) and others. The problem with instrumented vehicles is the small observation rate of the whole traffic and the accuracy of the measurement equipment. To overcome these drawbacks, Hoogendoorn et al (2004) collected data by remote sensing with sequences of high-resolution pictures from a helicopter and the Next Generation SIMulation Community (NGSIM) placed video cameras along motorways in the USA to record trajectory files of individual vehicles (http://ngsim.camsys.com/).
With this data on hand, car following models can be calibrated and validated based on a bigger sample of datasets and for different states of traffic like free flow, starting congestion, congestion and resolving congestion. Ossen, Hoogendoorn & Gorte (2006) found that inter-driver differences can not be caught by different parameter settings alone. As their analysis showed, the driving styles of different drivers appear inherently different and that not each driver can be represented best by the same model with parameters adapted to individual characteristics.

4.3.3 Lane changing

Next to the longitudinal driving behavior, another important aspect is the lateral, or lane changing, behavior of drivers. Gipps (1986) proposed a framework for the structure of lane changing decisions in urban driving situations including the influence of traffic signals, obstructions and different vehicle types such as heavy vehicles. The model concentrates on the decision-making process considering the potentially conflicting goals and assuming a logical driver behavior. The model also considers the urgency of the lane changing maneuver in terms of the distance of the intended turn of the driver. The urgency of the maneuver is modeled through the drivers' gap acceptance and braking behavior (Hidas 2002).
Yousif and Hunt (1995) developed a microscopic simulation model for the investigation of lane changing behavior on multi-lane unidirectional roadways. The rules pertaining to the desire and the possibility to change lane are based on similar logic to that described by Gipps (1986). The assumption of the model is that if the available gap in the target lane is smaller than a given acceptable limit, no lane changing will take place. The main concern of the study is the relationship between lane utilization and traffic flow on dual-carriageway roads under normal flow

conditions (i.e. without incidents) and the model is adequate for this purpose. However, it could not produce realistic results when incidents or lane closures affect the flow conditions. These findings see to suggest similar conclusions found for the car following models: That not every scenario of geometry and density can be simulated best with one single model.

Barcelo et al. (1996) describe the AIMSUN2 microscopic traffic simulator developed for modeling real-time traffic management and information systems. The lane changing model is based on Gipps' model (1986). Although it is stated that AIMSUN2 can also model incidents, no information is given on how the model deals with lane changing under incident situations (Hidas 2002).

As opposed to the commonly used rule-based methods, a different approach is taken by Hunt and Lyons (1994) who developed a driver decision-making model for lane changing using neural networks. Their model works by assessing simple visual pattern-based input describing the driving environment around the vehicle about to change lane; it does not consider possible cooperation between drivers during lane changing.

One simulation model which explicitly addresses cooperative lane changing is MITSIM (Yang and Koutsopoulos, 1996), in which a courtesy yielding function is used to make space for a vehicle moving into the lane. While no information was found in the literature on forced lane changing, similar behavior was described in a recent study of gap acceptance behavior at roundabout entries by Troutbeck and Kako (1997).

A detailed description of the lane change behavior implemented in the simulation model SIMONE can be found in Minderhoud (1999). SIMONE distinguishes between mandatory and discretionary lane changes and defines explicit requirements for lane changes to the left or right separately. The microscopic simulator FOSIM (Dijker et al, 2005) handles lane changing separately for changes in the road geometry and the resulting weaving and merging actions. Ngoduy (2006) makes the same distinction of lane changes in his macroscopic model for lane changes at discontinuities.

4.3.4 Behavior patterns from microscopic measurements

The discussed literature showed that describing the driving task and thus human behavior is very complex. Psychological approaches are able to describe the driving task, but do have problems in the description of human interaction and existing mathematical approaches are not fully able to reproduce the individual driving behavior in a uniform, generic model either. Therefore, we describe drivers as reactive agents with individual capability to perceive their environment, make

decisions based on what they 'see' and take appropriate actions. This autonomous and partly unpredictable driver behavior leads to the emergence of traffic flow due to interactions between individual agents. Each agent perceives the actual situation with sensors, which gives the perception of the driver. These sensors are used in a belief network (BN) to calculate the actual belief state of the driver according to car following, lane changing and driving comfort. A Bayesian Belief Network (BBN) is a special type of diagram (called a graph) together with an associated set of probability tables. The nodes represent variables, which can be discrete or continuous and the arcs represent causal relationships between variables. The key feature of BBNs is that they enable us to model and reason about uncertainty, similar to models like neural nets, fuzzy control
In the BBN we model this by filling in a probability table for each node. To find the so called Node Probability Tables or NPTs, there are several ways of determining the probabilities of any of the tables. The beauty of BBNs is that we are able to accommodate both subjective probabilities (expert knowledge) and probabilities based on objective data.

If a node has more than one parent node in the graph, additional conditional probability tables are needed. BBNs make explicit the dependencies between different variables. In general there may be relatively few direct dependencies (modeled by arcs between nodes of the network) and this means that many of the variables are conditionally independent. The existence of unlinked (conditionally independent) nodes in a network drastically reduces the computations necessary to work out all the probabilities we require. In general, all the probabilities can be computed from the joint probability distribution. Crucially, this joint probability distribution is far simpler to compute when there are conditionally independent nodes.

Suppose the set of variables in a BBN is $\{A_1,A_2,\ldots,A_n\}$ and that parents(A_i) denotes the set of parents of the node Ai in the BBN. Then the joint probability distribution for {A1,A2,…,An} is:

$$P(A_1,...,A_n)=\prod_{i=1}^{n}P(A_i \mid parents(A_i)) \qquad (2)$$

with:

$P(A_1,...,A_n)$: joint probability distribution of $(A_1,...,A_n)$
$P(A_i \mid parents(A_i))$: conditional probability of A_i given parents(A_i)

Imagine a large net with many dependencies and nodes that can take on more than two values. Doing the propagation in such cases is generally very difficult. In fact, there are no universally efficient algorithms for doing these computations (the problem is NP-hard). This observation, until relatively recently, meant that BBNs could not be used to solve realistic problems. However, Rumelhart and McClelland (1986) discovered propagation algorithms that were effective for large classes of BBNs. With the introduction of software tools that implement these algorithms (as well as providing a graphical interface to draw the graphs and fill in the probability tables) it is now possible to use BBNs to solve complex problems without doing any of the Bayesian calculations by hand. The most widely used BBNs are the ones embedded in Microsoft's products as trouble shooter and technical support (Microsoft, 1995), and the Vista system, which is a decision-theoretic system that has been used at NASA Mission Control Center in Houston (Horvitz, 1992). Others are medical diagnostic (Wiggens, 2004), traffic monitoring (Huang et al, 1994) and the Bayesian automated taxi (Russel, 1997).

Dynamic belief networks allow reasoning in domains where variables take on different values over time. Typically, observations are taken at regular time slices and a given network structure is replicated for each slice. Nodes can be connected not only to other nodes within the same time slice but also to nodes in the previous or subsequent slice. As new slices are added to the network, older slices are removed. Before a slice is removed its influence is rolled up into the next slice by re-computing probability tables for certain nodes in that slice.

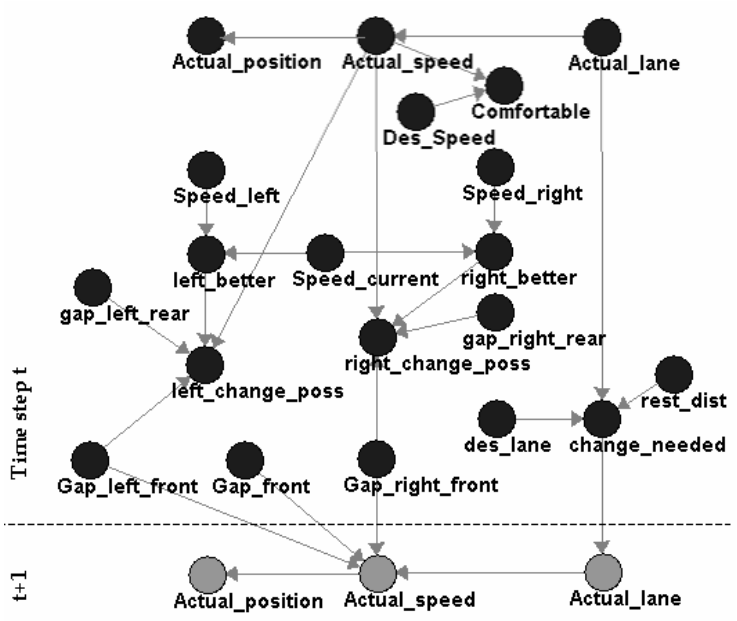

Figure 22: Structure of the BN for the believe state of a driver

Thus, evidence accumulated over time is always integrated into the current belief network model (Nicholson, 1992, Kjaerulff, 1993). The probability functions of the BN are findings from remote sensing and probe vehicle data and will change online if new measurement data is added. Due to this approach the model is able to adapt to different kinds of driving behavior, as long as there is measurement data available.

Based on the belief states the driver will take a decision. Therefore, an action node is added to the BN to create a decision network (DN). The decision is influenced by the belief states as well as by a utility function that defines the goal for each driver separately. The probabilistic network for the speed decision is generated from the existing data, which is available.

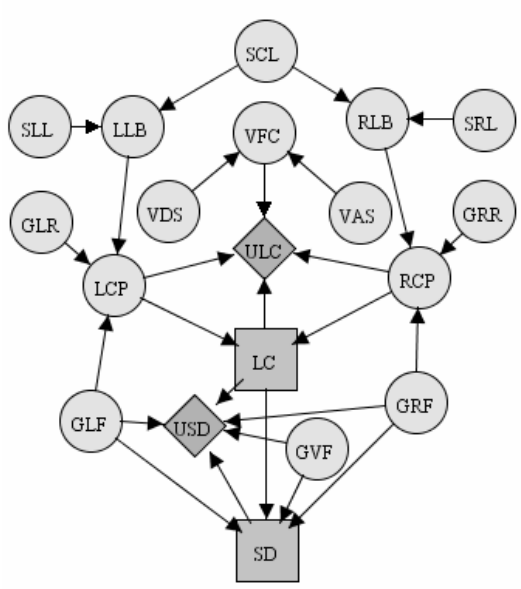

Figure 23: Structure of the decision network to determine the reaction of a driver

To build a BN means to identify the structure (directed acyclic graph) and the numeric parameters (conditional probabilities tables). This can be done in three forms:

- by an expert domain,
- automatically from data, or as a
- combination of these two.

Algorithms for structural learning are divided in search and scoring based algorithms (Heckerman, D., 1999 and Buntine, W., 1996) and dependence analysis algorithms (Cheng et.al., 2002, Pearl, 2000, Sprites et. al., 2000). Here we used the UnBBayes

learning tool (http://unbbayes.sourceforge.net/), which uses an algorithm for structural learning in BNs given a dataset using dependence analysis. It uses the XPC algorithm, an extension of the PC algorithm (Sprites et. al., 2000), developed by Fernandez (2004) to generate the BN. The probabilistic network is shown in Figure 22.

For the decision making process, the DPN is extended by an action node to a dynamic decision network DDN. Action nodes represent the reaction of a driver due to his surroundings. For the cellular automata model, these actions are the lane change and the speed change. All involved sensor nodes of a time slice are connected to these action nodes. These action nodes determine whether an action is possible and which options are available for the action. For the speed decision this is the acceleration, deceleration or continuing with the actual speed and for the lane change it is keeping the lane or a change to the left or right lane. Additionally the DDN gets a reward node for each time slice. These reward nodes represent the utility of the combined action of speed decision and lane change. While a lane change gives a high reward when changing lanes while the actual speed is lower than the desired speed the reward from the speed decision could be low because the actual speed on the other lane is lower than the one on the actual lane. With the combination of these two reward values, the final decision is made to maximize the reward. The decision process is not new, but instead of finding parameters for a model by fitting it to real world measurements, the measurements are the source of the model and the model will adapt to the measurements it gets fed. One can argue that the system resolves in a black box, but with this black box based on real world observations and learning from it, it functions similar to a neural network, a technique well established in traffic engineering. Given that behavior patterns on certain locations reoccur (e.g. at weaving sections or motorway ramps) this model allows self learning and tuning from measurements at this particular spot.

While MiOS is using this approach in a basic way it gives opportunities to tune the driving behavior in more detail. Adding a desired destination lane on the link and the distance to the end of the link could reflect the willingness of a driver to change lanes aiming for an off-ramp. Further possibilities would be to attach a tactical lane changing model (Webster et. al., 2005) or to incorporate route choice aspects like day to day learning (Viti et al, 2005) and en route information given to the drivers (Bogers et al, 2005).

The structure of the DDN is shown in Figure 23. The DDN is trained with a set of examples of belief-state/action pairs, which are taken from remote sensing data on Dutch motorways, US Highways and Japanese expressways. The remote sensing is

done with sequences of high-resolution pictures from a helicopter and with video detection from a high spot. The video data is processed and the results are vehicle trajectory files, which can be processed to determine situation dependant reactions of drivers. Details can be found in Hoogendoorn et al. (2004), Hoogendoorn, Thijs (2004) and the website of NGSIM – Next Generation SIMulation Community (http://ngsim.camsys.com/). On basis of this data an analysis of the vehicle trajectory data has been performed and is shown in Figure 24. The measurements cover observation from highway networks, starting with free flow conditions, turning into congested situation and the following resolving of the congestion.

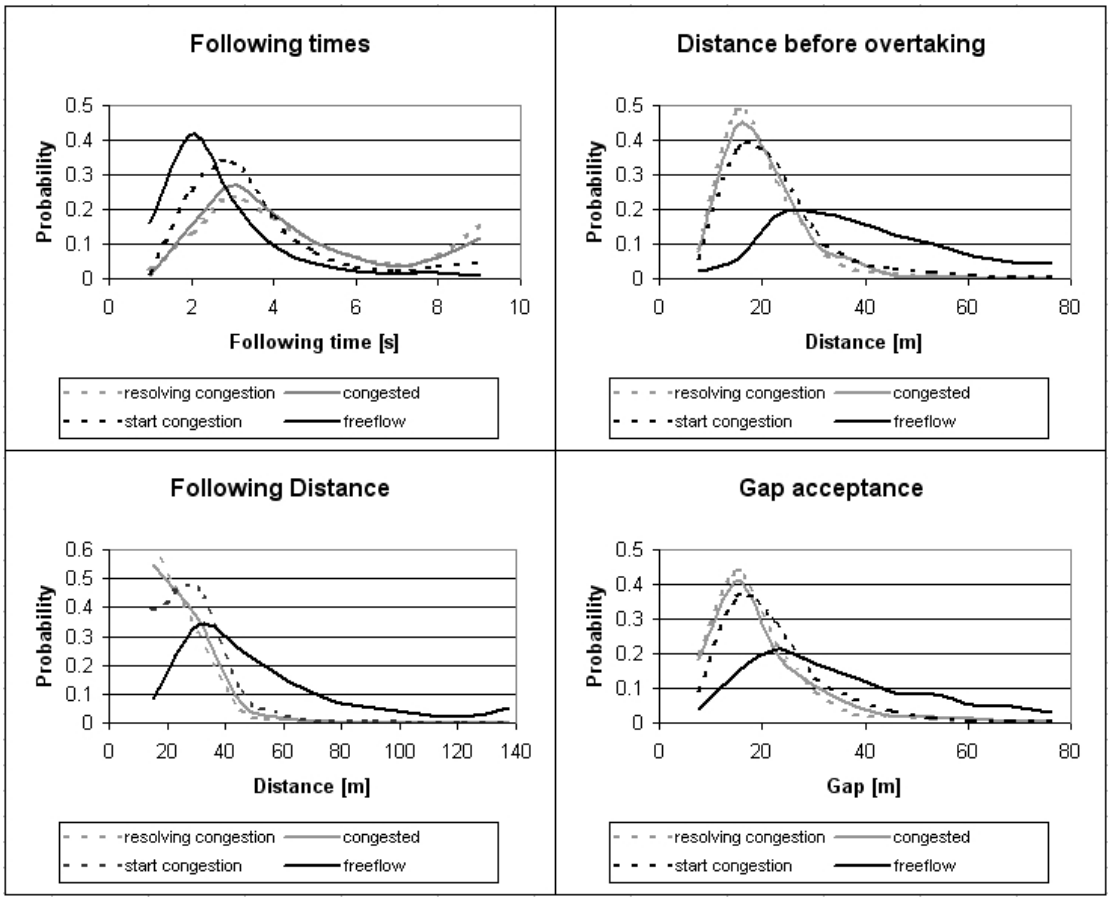

Figure 24: Findings from microscopic measurement data performed during the research

To find the distribution functions for the Bayesian network and the parameters for the decision network, the model is fed with different data sets, which have been extracted from measurement data. The data sets include the position and speed of the base car, the distances of the leading and following car, the occupancy of the neighbored and the changes in the driving state of the base car for the time step t+1. The given

training module of the UnBBayes package was used to input the data sets to the model. The outcome is a deterministic behavior model, which cannot directly be used for traffic simulation, since human behavior cannot be seen as deterministic at all. Drivers react differently on comparable situations, because of their perception of the situation and the personal characteristics.

To compensate this, noise is added to the perception of the driver according to the following distance and speed differences. Since the gaps are discretized by the cellular automata the noise is given in cells. The noise randomly adds an error $\varepsilon \in \{\varepsilon_{min}, \varepsilon_{max}\}$ to the individual perceived gap. The range of ε is dependant on the situation and defined as follow:

$$\varepsilon = \begin{cases} \in \{-2,2\}\ if\ \ v_{act} - gap > 2 \\ \in \{-1,1\} if\ \ v_{act} - gap \in \{1,2\} \\ \in = 0\ \ otherwise \end{cases} \tag{3}$$

These values have been chosen from by consulting experiment reports, which examined the perception errors in speed and distance during the driving task (Fildes et. al., 1989 and Treiber et. al., 2005). Therewith, the resulting behavior becomes stochastic one and can be used for traffic simulation. An online training with further measurements is at this stage not possible, but the values for the probability tables and the reward nodes can be changed offline and the driving behavior module can be exchanged in total.

4.4 Route Choice

4.4.1 Route choice models for microscopic simulations

It would be desirable to use a similar approach for the route choice than we already introduced for the driving behavior. However, real world data is not available in that amount, needed for such an approach. Therefore, a method based on graph theory has been chosen that is based on network conditions, which can be adjusted through traffic forecasts and the comfort of a driver, a parameter that might can be set for the individual driver when future observations allow an extraction of its distribution.

Traffic conditions in dynamic models are represented by a cost for each link of the network. These costs are mainly calculated on the basis of the average travel time experienced by the vehicles that covered the link in the previous time periods (Engelson et al, 2003). This allows vehicles to use the shortest path computed by taking into account the current conditions. Consequently, Wardrop's principle can be roughly respected. A sort of dynamic equilibrium is then obtained, even if the words "dynamic" and "equilibrium" seems somewhere in contradiction. Thus, it can be shown as a reactive or adaptive traffic assignment (Torday et al, 2004). The major difference between this model and real route choice behavior is that drivers usually have a more or less rough idea of traffic conditions evolution during a typical day (usually named historical profile) but, without information, they cannot know the exact conditions of a particular day as it is the case with the classical dynamic traffic assignment (DTA). It is particularly the case when traffic conditions are widely varying from one day to another which is mostly the case in complex urban networks. Hence, route choice capabilities of non informed vehicles can be considered as "too good" in a classical DTA model.
In addition, other difficulties may appear when large networks are used. In this case, the difference between the information used to assign a route to a vehicle at its entrance time and the ones it will experience arriving at a sensible point situated faraway (in time) from the entrance can lead to a situation far from the equilibrium statements. To put this problem right, micro simulators usually provide the possibility of rerouting the cars after a certain period in order to take into account new traffic conditions (Torday et al, 2004).

4.4.2 Route choice in MiOS

MiOS also introduces link costs to determine the route choice of the individual drivers, similar to the standard route choice models. Therefore, the network is translated to a directed graph G := (V;R), where V is the set of vertices and R $\subseteq$ V x V is the set of ordered vertex pairs (edge set), to use formal graph theory logic to find the shortest paths.
The generated graph is antireflexive (no vertex with the same start and end node), asymmetric and cyclic. The routes between origin *o* and destination *d* are represented by the set a_{od} containing links between vertex *o* and d. The minimum of the length of the paths contained in a_{od} is chosen as the weight z_{od} of the path set a_{od}. If the path set a_{od} is the zero set 0_W (no path between *o* and *d*), then $z_{od} = \infty$. If the path set a_{od} is the unit set $1_W(o=d)$, then $z_{od} = 0$. If the path set a_{od} is neither the zero set 0_W nor the unit set 1_W, then z_{od} is a non-negative real number. Thus, the weight mapping is, according to Pahl et. al. (2001), defined as follows:

$$\begin{aligned} &\textit{zero element (no path)}: && 0_Z = \infty \\ &\textit{unit element } (o = d): && 1_Z = 0 \\ &\textit{element}: && z_{od} \in \Re_0^+ \;\; \textit{for } a_{od} \notin \{0_W, 1_W\} \end{aligned} \tag{4}$$

The elementary weight matrix **Z** of the graph contains link based weights z_{ik} and gets updated at a user defined time-step. We define the weights by the length of the link l_{ik} from vertex i to vertex k, the actual average speed u_{ik} of the amount of V vehicles on that link, the maximum speed $u_{max,ik}$ and the recommendation parameter Θ, which allows to incorporate the effect of external route guidance:

$$z_{ik}(t) = \frac{l_{ik}}{\sum_V u_{ik}(t) + u_{\max,ik}} * (V(t)+1) * \Theta(t) \tag{5}$$

The first two factors represent the average speed of all cars on the link. Additionally the maximum allowed speed is added, which represents the link status. If there is no car on the link, the average speed is the maximum speed, if there are a few slow cars, the average speed remains higher and if the traffic is very dense on the link the maximum speed becomes irrelevant. Note that in this equation we use space mean speeds.

Parameter Θ distinguishes the different road types and takes in-car route guidance systems and en-route traffic information into account. With parameter T for the road type α as an indicator for in-car systems and β as a factor for the willingness to follow the advice, Θ can be described as:

$$\Theta = (1 - \alpha \cdot \beta) * \mathrm{T} \tag{6}$$

with:

$\alpha \coloneqq \textit{fraction of equipped cars}$,

$\beta \coloneqq \textit{fraction of drivers following the advice}$,

$$T := \begin{cases} T^{motorway} & for\ motorways \\ T^{urban} & for\ urban\ roads \\ T^{city} & for\ inner-city\ roads \end{cases}, T \in \{0, 15\}$$

Parameter T is used to represent the attraction of certain road types, meaning that a motorway with a slightly higher travel time than an urban road alternative can still be more attractive to use. However, if the car is equipped with route information and follows that advice, the influence of the road type T is neglected. So, a driver in an equipped vehicle, willing to follow the guidance selects the advised route, which is assumed to be the shortest. The model is not calibrated and based on personal experience and survey. Breaking down the route decision to what the driver perceives leads to intuitive decisions and is assumed to be face valid.

When a vehicle is generated, it is assigned to the shortest route between its origin *o* and its destination *d* in an empty network. This can lead to a flip-flop effect between the alternative routes between an origin and destination which is not realistic. One approach to overcome this phenomenon is to insert a threshold before drivers start to change their routes through the network. In MiOS a comfort factor is used, representing the fact if a driver is satisfied at each moment during his journey. This factor and its distribution could in future be derived from observations. If the driver does not feel comfortable anymore, which means that the comfort factor $C_D < 0$, he is trying to find another route.

Let be:

$u_{a,v}(t)$ the actual speed of driver [km/h]
$u_{d,v}(t)$ the desired speed of driver [km/h]
$v_{r,v}(t)$ the number of cars on link r [vehicles]
$c_{r,}$ the maximum amount of vehicles that can be placed in congestion on link r [vehicles].

To determine the desired speed of a driver, remote sensing data has been used. Data from non congested conditions has been analyzed to find a distribution of desired speeds around the speed limit, given on the road stretch. For further details see the Appendix A.

Then we define comfort factor C_D of driver D as:

$$C_D = \frac{u_{a,v}}{u_{d,v}} - \frac{v_{r,v}}{c_r} \qquad \text{, with } C_D \{-1, 1\} \tag{7}$$

The comfort factor is 1 when the driver can drive with desired speed in an empty network. The factor decreases with an increase of the traffic volume until it reaches 0 when the vehicle is on a jammed link. Between these values lies an individual threshold for each driver from which he or she does not feel comfortable anymore on the chosen route.
If a driver does not feel comfortable anymore the system triggers the model to calculate a new route. That means, if P^0_{od} is the shortest path between origin *o* and destination *d* in the empty network and P_{od}, assumed to be the perceived shortest path in the network, that:

$$\begin{aligned} &\textit{if } C_D > 0 && P_{od} = P^0_{od} \\ &\textit{else} && P_{od} = \min P_{od} \textit{ in } G \end{aligned} \tag{8}$$

The actual shortest route at instant *t* is calculated with the *Floyd-Warshall algorithm* (Corman et. al.). This algorithm is based on the *Dijkstra* algorithm. It should be noted, that depending on the network size and network structure, other algorithms can be more efficient, but the choice of shortest-path algorithm for a particular problem will involve complex tradeoffs between flexibility, scalability, performance, and implementation complexity (Foster, 1995). The *Floyd-Warshall algorithm* is an algorithm to solve the all pairs shortest path problem in a weighted, directed graph by multiplying an adjacency-matrix representation of the graph multiple times. The time complexity is $\theta(V^3)$. The computational load of this algorithm scales with the amount of nodes and links in the network and is not dependent on the number of vehicles as other modules like the simulator or the driving behavior. If however the network size exceeds the computational power of the machine performing this task, an optimization of the graph *G* can result in a faster processing.

4.5 OD estimation and prediction

Origin-destination (OD) demand matrices estimation and prediction is important for traffic control and management policies and even more important for real-time monitoring and control of road networks. This section gives a short overview of some

existing methods for OD estimation as an overview and describes the implementation of OD estimation and prediction in MiOS.

4.5.1 Existing OD estimation methods

Dynamic OD estimation builds on static OD matrix estimation. Considerable work on static OD estimation has bee done by Abrahamsson (1998), Cascetta and Nguyen (1988), Cascetta (1984), and Cascetta (2001). Hazelton (2000) paid special attention to the statistical aspects of OD estimation and a bi-level programming approach has been introduced by Florian and Chen (1991) and Yang et al (1994).
A framework for general dynamic OD matrix estimation is given in Cascetta et al (1993). An estimation approach based solely on traffic counts for general networks is given in Sherali et al (2001). For the special case in which the entire network on which the OD matrix is to be estimated several sources are available such as Kremer and Keller (1987), Nihan and Davis (1987), Bell (1991), and Van der Zijpp (1996).
Lindveld (2003) categorized existing OD matrix estimation methods and distinguishes them along the following attributes:

- Time-frame: Online estimators must work in real-time and therefore have limited data available while off-line estimators have all needed data available.
- Problem formulation: Simultaneous estimators estimate the whole dynamic OD matrix at once, while sequential estimators estimate time slice by time slice.
- Model approach: Either purely statistical with scant attention paid to behavioral aspects of the underlying transportation phenomena, or the approach incorporates these behavioral elements (especially departure-time choice).
- Type of network: The network can either be closed (all entries and exits are continuously monitored) or a general one.
- Route alternatives: The presence or absence of route alternatives in a network determines whether the problem of relating aggregate traffic observations to individual OD flows has a spatial component.
- Estimation method: One can distinguish between estimation methods that exclusively rely on traffic counts and those that can incorporate other data sources.
- Apriori matrix: Another characteristic is the presence or absence of an apriori matrix and whether it is used explicit (apriori matrix given) or

implicit (e.g. selection of the most likely matrix through the Maximum Entropy principle)

Table 2 lists the approaches to dynamic OD matrix estimation according to the above characteristics.

	Time frame	Problem formulation	Model approach	Type of network	Route alternative	Estimation method
Kremer & Keller (1987)	online	sequential	statistical	closed	no	counts
Camus et al. (1997)	online	sequential	both	general	yes	counts
Ashok (1996) Sherali et al. (2001)	both	sequential	statistical	general	yes	counts
Van der Zijpp (1996)	offline	simultaneous	statistical	closed	no	counts
Tavan & Mahmassani (2001)	offline	simultaneous	route choice	general	yes	counts
Lindveld (2003)	offline	simultaneous	route choice & departure time choice	general	yes	group

Table 2: List of approaches to dynamic OD matrix estimation (Lindveld 2003)

Lindveld (2003) also mentions the variety of objective functions from the methodology correct Likelihood function (ML) and Minimum Information estimators (van Zuylen, 1983) to the quadratic and Maximum Entropy methods (van Zuylen, 1983). Since OD matrices can hardly be measured (except for closed networks with vehicle identification at the entry and exit points) it is not easy to assess the existing methods with real world data. More can be found in Mahmassani et al (2005). MiOS does not intent to introduce a new way of OD estimation and prediction. Instead it offers a very simple method for small networks (described below), and additionally an interface to use an external OD estimation and prediction tool (see appendix).

4.5.2 Implementation in MiOS

The vehicle input in MiOS is a set of static OD matrices for different time periods, which are corrected during runtime by factors. These factors are determined by given online traffic measurements. These measurements come in every minute from inductive loop detectors on the motorways and intersections as well as from video camera detection. The volumes of the simulation are compared to the measured values of the real world and the difference determines the change of the OD matrix. For the ongoing estimation of the traffic situation this difference should not go beyond a given threshold. If this is the case, the estimation will be restarted to assure the quality of the shown image. The prediction is always based on the newest measurements and is so strongly dependent of the frequency of the incoming data.
In MiOS the dynamic OD matrix is based on four static OD matrices for different times of the day (morning, afternoon, evening, night), which have different characteristics. Depending on the time of the day, one of these matrices is multiplied with a factor vector, depending on the online traffic measurements. That means, that the flow f_{ij} between the origin i and the destination j is determined from the static OD value h_{ij} and a vector α, which includes the estimate m_i of the production of origin i.

$$f_{ij}(t) = \alpha_i(t) \cdot h_{ij} \ , \qquad with \ \ \alpha_i(t) = \frac{m_i(t)}{\sum_i h_{ij}} \tag{9}$$

The static OD matrices are considered to be given, based on surveys in the research area. The vector α is calculated in two different ways, depending on the available measurements. If measurements are available at the entry or exit point of the model, than the aggregated measurement value is compared to the values of the production in the static OD matrix. Based on the difference, the factor for multiplying the static values is calculated. If no measurement is available, the factor is determined by a combination of the calculated factors. This should only be done for entries with a minor input, because the error of this input cannot be controlled.
The prediction follows the same concept as the estimation, but instead of only the online traffic measurement, the vector α is also based on a historical database of travel counts. That means, that historical data is used to predict roughly what will happen in the future. This prediction is corrected with a factor, considering the difference between the actual measurements and the measurements in the past:

$$f_{ij}(t+n) = \beta_i \cdot h_{ij}(t+n) \quad , \qquad with \quad \beta_i = \frac{\sum_{x=0}^{15} m_i(t-x)}{\sum_{x=0}^{15} \sum_i h_{ij}(t-x)} \tag{10}$$

The factor β is determined by the difference between the measurements and the actual traffic counts from the last 15 minutes. So if the actual measurements are lower than the comparable historical data, it is assumed that this fact will not change over the next 30 minutes.

4.6 Discussion

This chapter described the methodology of the simulation components of MiOS. We extended a cellular automaton for heterogeneous traffic and a better representation of speed differences and gap sizes. The new proposed route choice model uses link impedances based on space mean speeds and by defining a comfort factor to trigger the willingness of changing routes for individual drivers allows a better calibration of observed situations. Although these models could not empirically be validated, they have face validity in the meaning that their characteristics are consistent with what one should intuitively expect. New is the driving behavior model based on microscopic measurement data that can be extended further to reflect the complexity of human behavior. The OD estimation and prediction is very simple in its present form, but can be replaced by another module, just as all other simulation components. With these components it is possible to simulate a traffic network with given structure and demand. All above models give room for improvement and need further development. The aim was to follow ideas motivated by discussion and insights from the assessment of microscopic simulation models. As mentioned in the beginning of this chapter, the main purpose of the developed models is the testing of the system architecture proposed since it is meant to be a platform for testing new algorithms and models under development.

Due to the modular structure of MiOS the components need to be connected. Therefore, the next Chapter describes the combination of the MiOS simulation modules with the FrOnT framework. It describes the information flow between the two applications and shows the opportunities of the coupled system.

5 Using FrOnT and MiOS

After describing the methods behind the systems FrOnT in Chapter 2 and MiOS in Chapter 4, this chapter gives an introduction of the use of both systems for a microscopic online simulation. First we describe the necessary actions needed to run the system and then describe the steps of a simulation with MiOS by inputting a network, defining controllers and OD matrices and finally running the simulation. The end of the Chapter shows the possibilities for simulation outputs and there further use.

5.1 Getting started

There are two ways to get the system running, independent from the runtime environment (Windows®, UNIX®, Linux®, MacOS®).

MiOS Basic
In the basic version, MiOS can be started directly and the startup procedure will start a set of agents consisting two control agents, ten collector agents, one communication agent and one recovery agent automatically. In a second step MiOS will start the following services and register them:

- Traffic control server
- OD estimation server
- Driving behavior server and two clients

With this set of agents and services MiOS can run on a single computer. The clients for traffic control and OD estimation are recognized during runtime and don't need to be defined during the startup procedure. The communication and recovery agent are started for an increased robustness of the system. Even though there is no communication to other computers in the network, the communication agent will mirror the network states in key frames and the recovery agent will store these in an object file to the hard disk so that in a case of malfunction the system can restore from the last key frame automatically.

MiOS extended

The extended version of MiOS requires a runtime script for startup. In this script the user defines the basic agents to use with preferable locations and the IP addresses of the computers allowed to join the FrOnT system. Even though this is in contrast to the design of FrOnT as a peer to peer system, security issues are nowadays demanding some precaution to overcome an abuse of the system. The runtime script looks as follows:

```
*****************************************
FrOnT version 0.9.8.12
*****************************************
IP deny table:(Security Blacklist)
2
161.12.1.*
161.48.*.*
*****************************************
IP allowed table:
4                (number of connected computers)
130.161.12.206   (IP address first computer)
130.161.8.*
130.161.12.205
130.75.108.*
*****************************************
Collector Agents:
15
130.161.*.*
*****************************************
Control Agents:
5
130.161.*.*
*****************************************
Communication Agents:
7
130.161.*.*
*****************************************
Checksum: #27
FrOnT version 0.9.8.12
*****************************************
```

During the startup process FrOnT will create a table of denied and accepted IP addresses and tries to start the agents on the given locations. The “*” is used as a

wildcard and can be every number between 0 and 255, to allow an easy inclusion of a whole network in one step. To allow changes in the set up of the system, the runtime script is read every 15 minutes to detect changes, so that the server list can be shrunk or extended during runtime, without halting the system. Errors in processing the runtime script are displayed in through the network viewer of MiOS.

5.2 Simulating with MiOS

After the startup of FrOnT and MiOS the network viewer and network editor of MiOS are displayed (see Figure 25). This main window of the application provides the user with the possibility to create new networks or to load an existing network into the simulator.

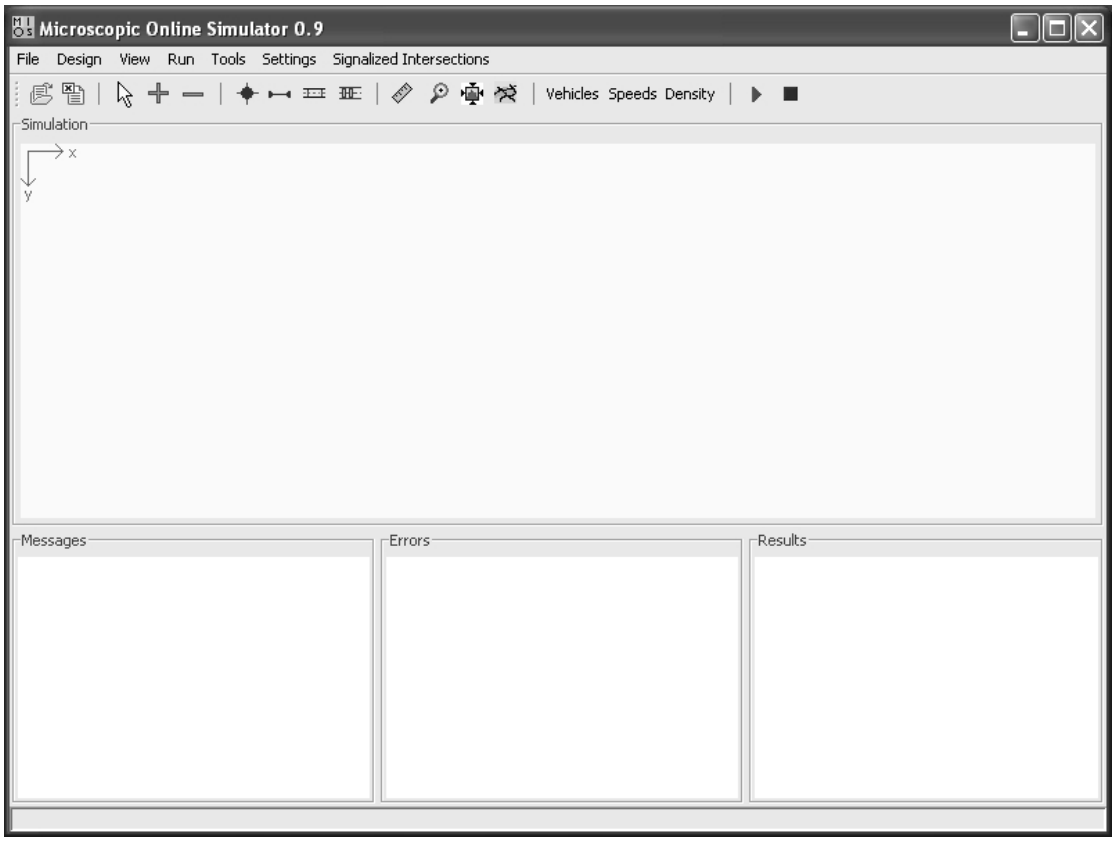

Figure 25: Screenshot of the MiOS editing and visualization tool

In the following we are going to describe the steps of creating a network and show the capabilities of the editor in the actual version. It is noted that MiOS is still a prototype

and while providing the fundamental user interface, a set of "nice to have" features are still missing, which would increase the user friendliness.

Starting a network from scratch usually involves a background of the simulation area, onto which the network is created. MiOS allows the input of several graphic formats, such as JPEG, PNG, BMP, and TIF. Technical drawings based on AutoCAD© have to be converted. In Figure 26 we show the network editor window with a loaded background taken from GoogleEarth™, showing the motorway A13 along the city of Delft. Onto this background the nodes and links of the network are created. For backgrounds exceeding the size of the screen it is possible to zoom in and zoom out (View menu) as well as to center a point of the network by clicking into the background.

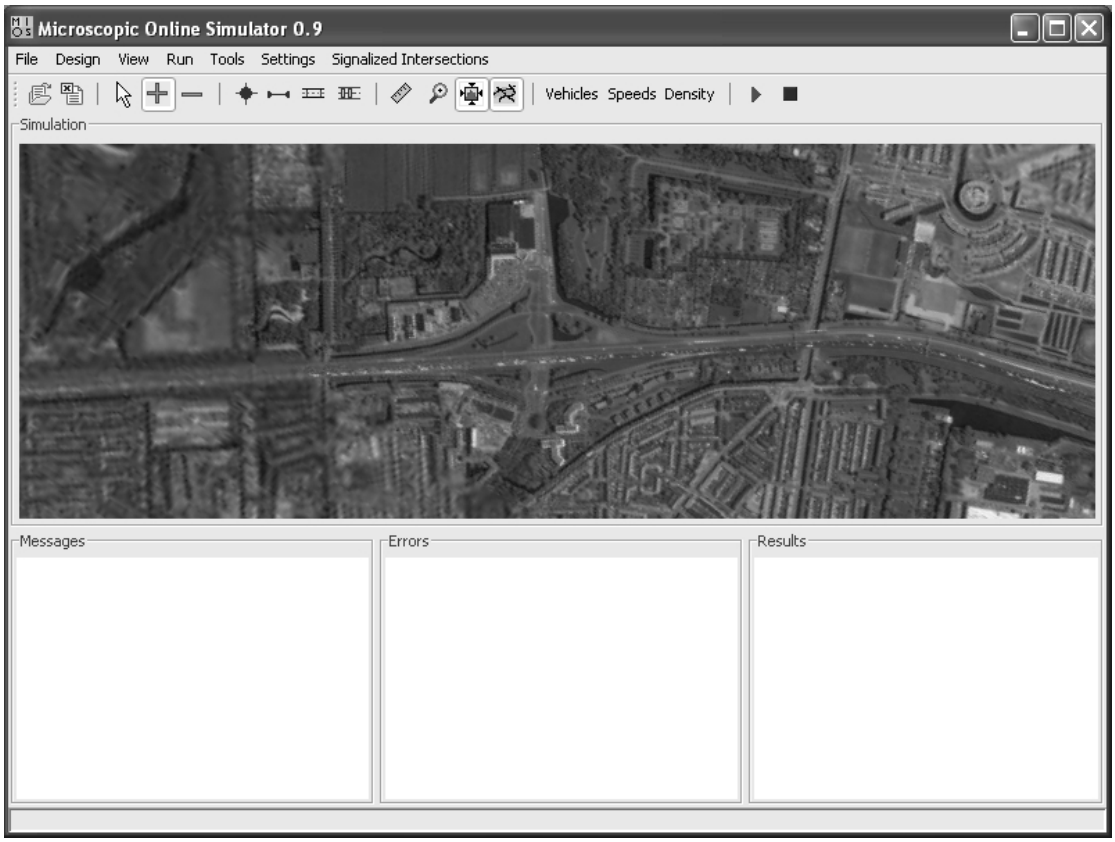

Figure 26: MiOS with a loaded background picture (GoogleEarth™)

After selecting the "plus" and "node" symbol for the toolbar or selecting "Adding node" from the design menu, nodes can be added to the network by clicking the left mouse button in the right position.

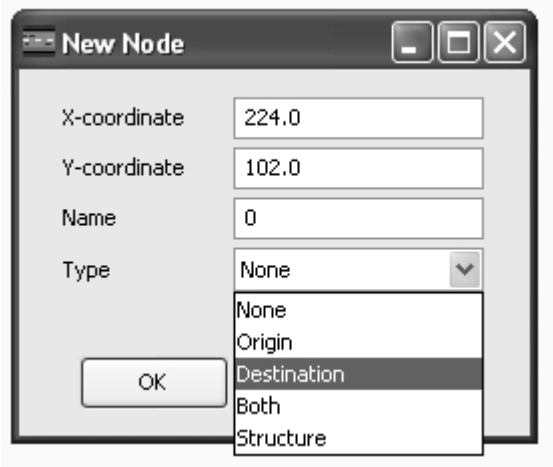

Figure 27: MiOS input dialog for adding a node

The input dialog for creating a node appears on the screen (see Figure 27) and shows the coordinates of the node in screen coordinates. The user has now the possibility to change the automatic generated name of the node (by default the internal node ID) to a number or string and to select a node type. The types of a node are:

- None (intermediate nodes)
- Origin (traffic production node)
- Destination (traffic attraction node)
- Both (sink and source for traffic demand)
- Structure (possibility for creating several nodes)

While the first four node types are straight forward, the structure selection will create a new dialog to define the structure in more detail (see Figure 28).

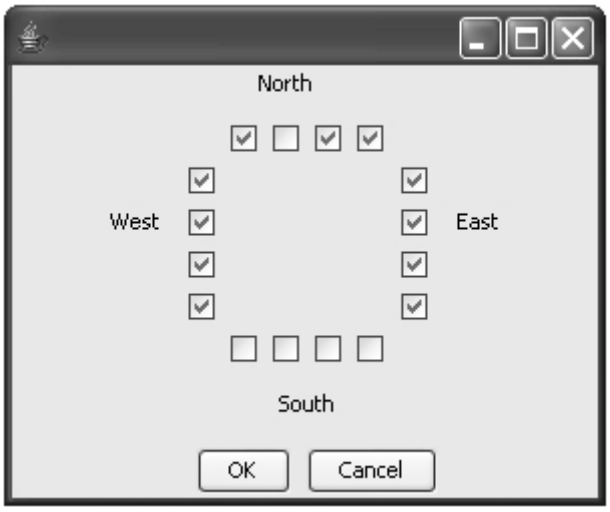

Figure 28: Dialog for defining a node structure in MiOS

The dialog displays a set of 16 node locations referring to a standard 4-leg intersection with its approaching and leaving links. By selecting the check boxes it is easy to create a four leg intersection at once. The nodes created by a structure will be by default of the type “none”.
After the nodes have been created the attributes can be changed by double clicking onto the node to open the node dialog again. A single click will highlight the node and allows moving the node by dragging it with the mouse over the screen. For a better visibility of the nodes, the color can be changed by clicking the right mouse button in the node selection mode. This will open a window with a color scheme to change the color settings.
In a next step the links are created. A link is defined by a starting node and an end node. MiOS is then automatically calculating the length of the link according to the coordinates. Note that the length is based on screen coordinates, which requires a re-scale of the whole network later on or a manual editing of the value for the link length. This is possible in the dialog for creating a link (see Figure 29).

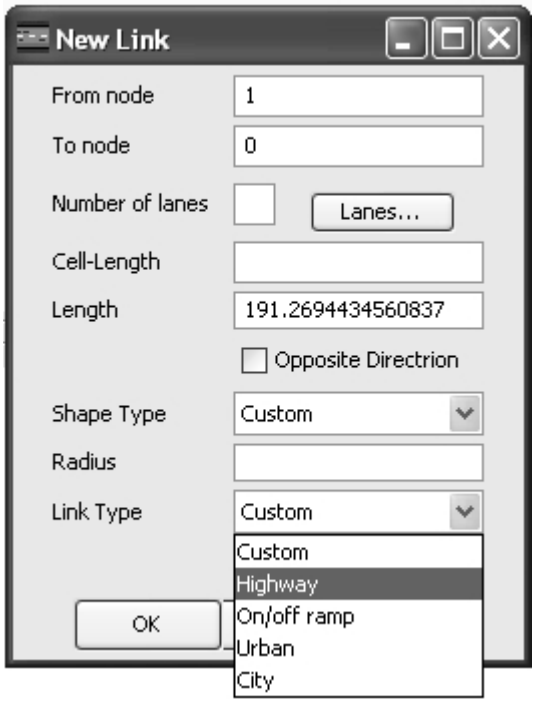

Figure 29: Inserting a link in MiOS

Next to the link length and the corresponding nodes, further definitions are possible in the dialog. MiOS is just using the information of the lanes, the length and the link type, which is used as an indicator for the route choice model. Information such as cell-length, shape and radius are meant for macroscopic simulators if used in the framework.

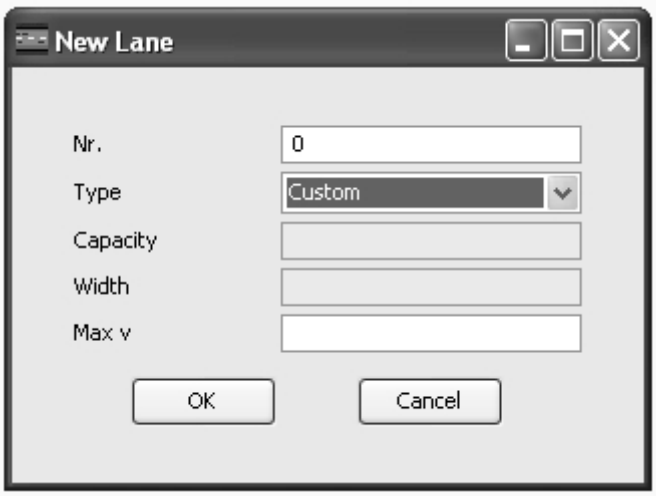

Figure 30: Dialog for adding lanes to a link in MiOS

To finalize the definition of the link, lanes have to be added to the connection. Then "Lanes…" button opens the dialog for the lane definition (see Figure 30). The lane number is auto generated and not editable. The values for type, capacity and width are again added to support the usage of other simulators, while MiOS in this prototype uses just the maximum speed, given in km/h. The maximum speed is defined lane wise to allow the speed limits per lane separately. After all lanes are defined the systems goes back to the link dialog. If a link is bi-directional the user can create the reverse link by checking the "Opposite Direction" box to avoid another link input. The values will be copied to a new link with the start and end node of the link switched.

Following these steps the basic network, consisting of nodes, links, and corresponding lanes is created. Figure 31 shows an example of the A13 network created on the background.

Figure 31: Correct way of modeling motorways in MiOS

Due to the limitations of the route choice algorithm it is essential to create a motorway stretch separately for both directions. Since the route choice allows U-turns, a congested motorway would otherwise lead to vehicles changing the direction on a motorway without using on- and off-ramps.
In the following we describe additional components of the network editing, such as controllers, detectors and measurement locations. All these components are attached to a link which should be kept in mind during the network construction. This means, that positions of traffic controllers, detectors and measurement points should be marked with a node. In this way the location of the component to add will always be at the front or end of a link, which is necessary for the further modeling of the network.
As a next step we add traffic lights to the network, located at the intersections at the end of ramps to the motorway. To add a controller, the link on which the controller is placed needs to be selected by giving the start and end node of the link. This will open the "Link selection" dialog, showing the given values of that link. At the bottom of the dialog an "add Signal" button is located. By pressing this button, the controller dialog is opened (see Figure 32).

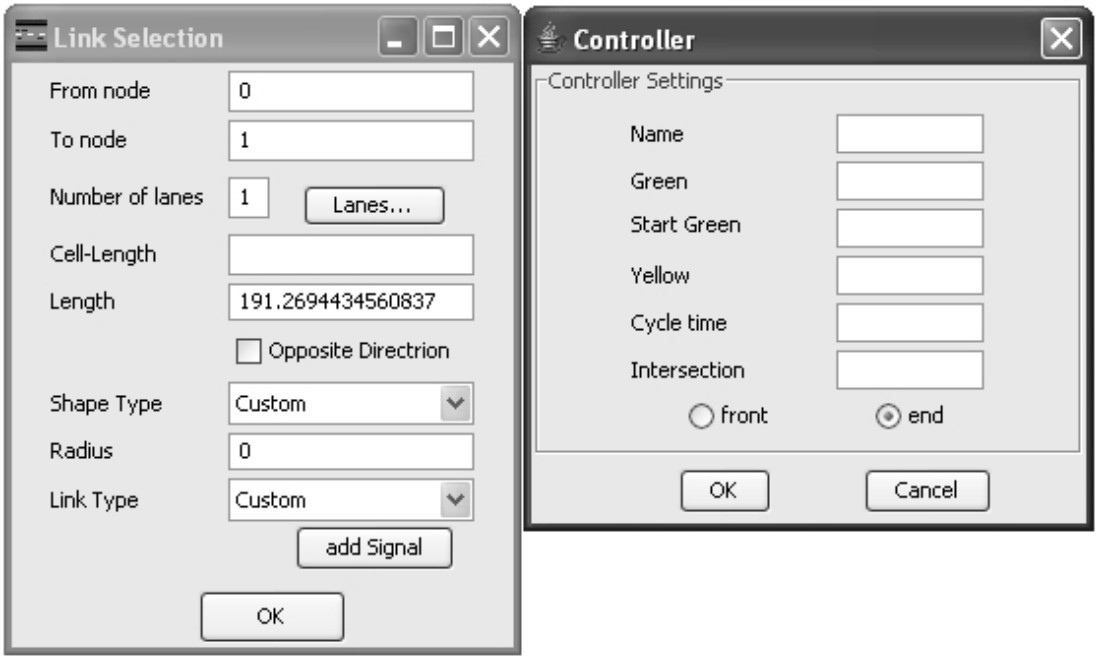

Figure 32: Adding traffic controllers to a link

The settings allow the user to define the name, the traffic light cycle of the controller, and offer the possibility to link the controller to a certain intersection. This data is used by the internal traffic controller of MiOS. If the intersection name however corresponds with the intersection names controlled by an external controller and its client is connected to MiOS the values are overwritten by the external traffic

controller. At the bottom of the dialog the user defines the position of the traffic light on the link (front or end). Every traffic controller is automatically equipped with detectors to detect the presents of a vehicle. However, if the user wants to place additional detectors in the network they have to be defined separately. Therefore, the mode is switched to “Create detector” and by selecting a corresponding link, a detector is placed in the beginning of the link.

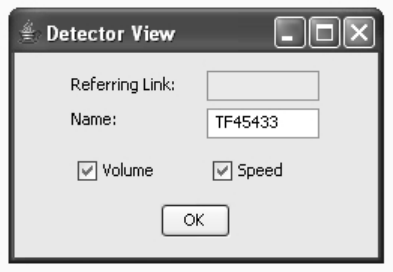

Figure 33: Adding detectors to the network

The input dialog for a detector (see Figure 33) additionally offers the user to define a name, which is independent from the internal ID used for the simulation and to chose the values to be measured (speed, volume).
In a simple manner measurement locations are built into the network. Measurement locations are used for the input of online measurement data, which are placed on a link. Vehicles are then created at the starting node of the link and enter the network. Even though the vehicles are added at the node, the input is defined on the link to include the direction of the detected vehicles. There are two different ways of using these measurement locations. When defined as offline these locations get connected to a data file which includes the measurements per minute while in the online use, they connect to a collector agent and receive the data according to the actual time. The mapping between the location and the collector agent is done by FrOnT. If “online” is highlighted the “…” button opens a list of all surveyed collection points be the agent society, while in the offline mode it opens a “File open” dialog.

Figure 34: Adding measurement locations used for online input to the model

With these steps, the creation of the network is finalized. MiOS saves the network in plain text files to allow an outside editing and import to other applications. The file contains a list of nodes and their location, followed by the list of links with included information for controllers, detectors and measurement locations.
To simulate the network it is further needed to input an OD matrix. Depending on the simulation purpose there are various ways to input the traffic demand:

- Static OD (constant demand)
- Base OD plus variation file (time varying demand synthetic)
- Base OD matrices plus offline measurements (time varying demand based on historical measurements)
- Base OD matrices plus online traffic counts (time varying demand based on real-time measurements)

For all cases a base OD matrices is needed. From the "Run" menu the topic OD matrix can be selected, which shows an OD table based on the defined origin and destination nodes (see Figure 35).

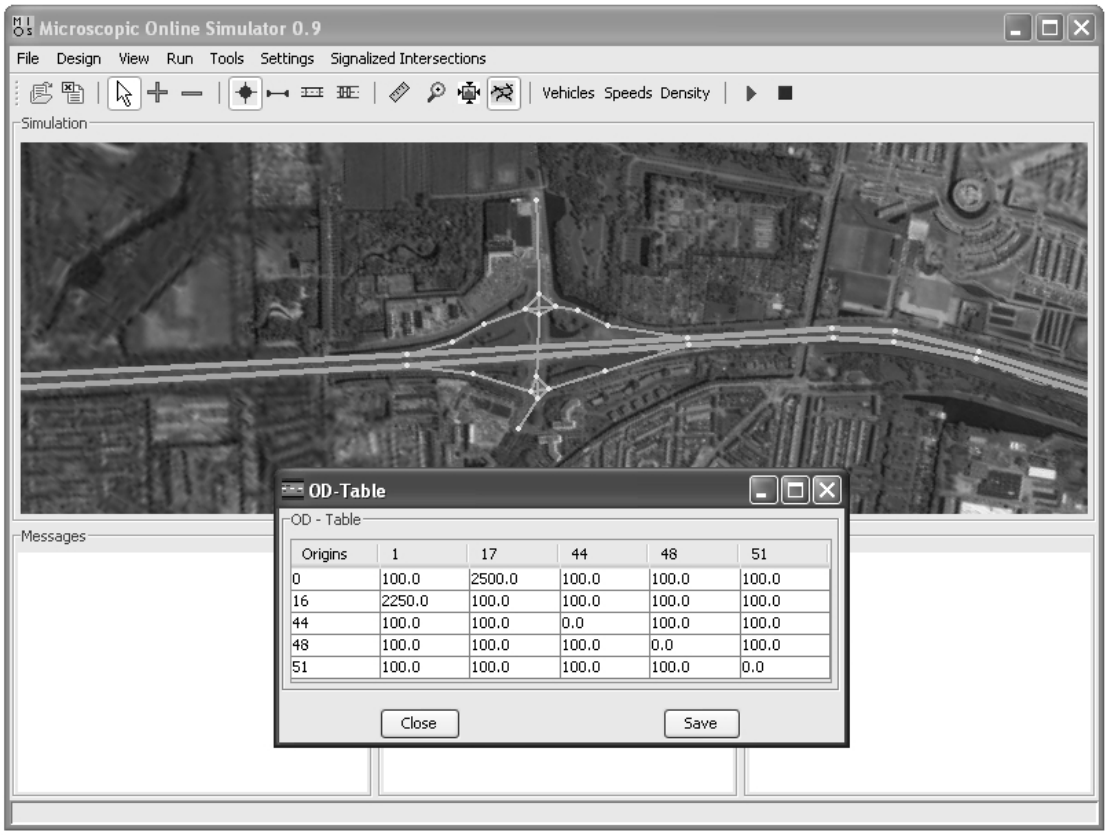

Figure 35: Definition of the OD table for the network

The base OD matrices are filled in traffic demand per hour. If no measurement points are defined in the network the simulation will search for a variation file called

"*od_variation.mios"* in the same folder as the network description file. If this file is found, MiOS will change the OD matrix every minute by multiplying each OD pair with the according value of the file. Such a variation file looks like:

```
*****************************************
FrOnT version 0.9.8.12
*****************************************
10
0.2 0.1 0.7 0.5 0.5 0.8 0.1 0.0 0.6 0.5
0.1 0.4 0.6 0.4 0.6 0.7 0.1 0.0 0.8 0.3
0.3 0.5 0.5 0.5 0.7 0.6 0.1 1.0 1.0 0.7
0.4 0.7 0.6 0.5 0.9 0.5 0.3 1.0 1.0 0.9
0.5 0.9 0.5 0.8 1.0 0.6 0.3 1.0 1.0 0.9
0.6 1.0 0.7 0.8 1.0 0.5 0.8 0.0 0.9 1.0
0.3 0.5 0.5 0.5 0.7 0.6 0.1 1.0 1.0 0.7
0.5 1.1 0.8 1.0 0.8 0.7 1.0 0.0 0.6 1.0
*****************************************
Checksum: #12
FrOnT version 0.9.8.12
*****************************************
```

The number of the first row gives the amount of OD pairs, followed by the variation parameters per OD pair in a row per minute. If this file is not present in the network folder, MiOS is starting a static simulation with a fixed demand given by the OD matrix.

Figure 36: Tools for scaling and enumeration of the network

Recalling a note from the link creation, the user should ensure to re-scale the network from screen coordinates to real distances. The "Tools" menu offers this possibility. By

selecting “Rescale network” (see Figure 36), the user can input a multiplier for all distances in the network, which can be found by checking the screen coordinates for a known distance on the background. Further the menu allows a new enumeration of links and nodes. This function eliminates unused node and link IDs internally and could be useful before exporting the link and node list to another software tool.

Now the network is ready for the simulation. To control the simulation the user further has the possibility to change the setting for the route choice model, described in Chapter 4. Through the “Settings” menu the user gets access to the route choice input dialog (see Figure 37) in which he can define the impedance of a link based on its type as well as the penetration of equipped vehicles and the information acceptance in percent. Ramps get the default impedance of 999.0 to avoid that vehicles during congestion take an off-ramp and immediately the next on-ramp again as it can be found in microscopic simulation models.

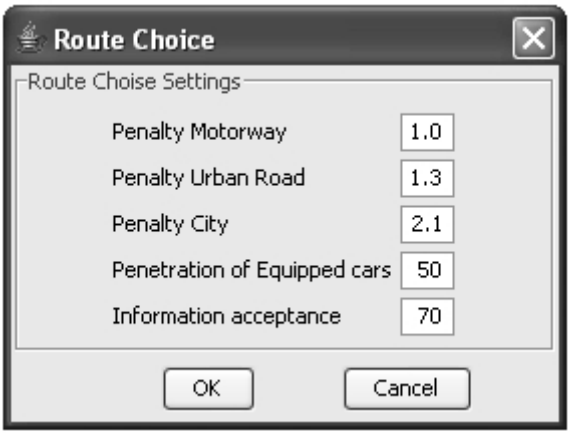

Figure 37: Route choice settings for the simulation

Further settings refer to the refresh time for the simulation. The “Refresh rates” dialog (see Figure 38) allows adjustments of the internal timer settings of the simulation. There are four values to set:

- Display: refreshing rate of the visualization of the network in the viewer window
- Route choice: interval in which the shortest path algorithm is run for the entire network
- Evaluation: interval in which average speeds and densities per link are calculated for displaying
- Simulation speed: adjusts the simulation speed for better visual checking of the traffic state.

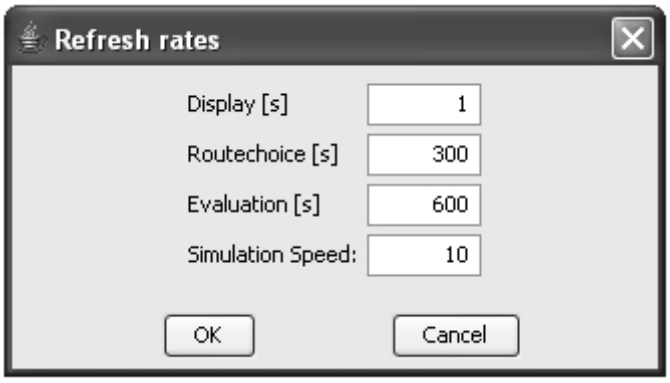

Figure 38: Timer settings for the simulation

With all these values set or left as default the simulation can be started, by using the "Play" button of the toolbar or using the "Run" menu. Then the network modeled gets translated into a simulation network and the software links are set to the agent society and external tools for traffic control and others. The nodes disappear and the links are substituted with road images (see Figure 39).

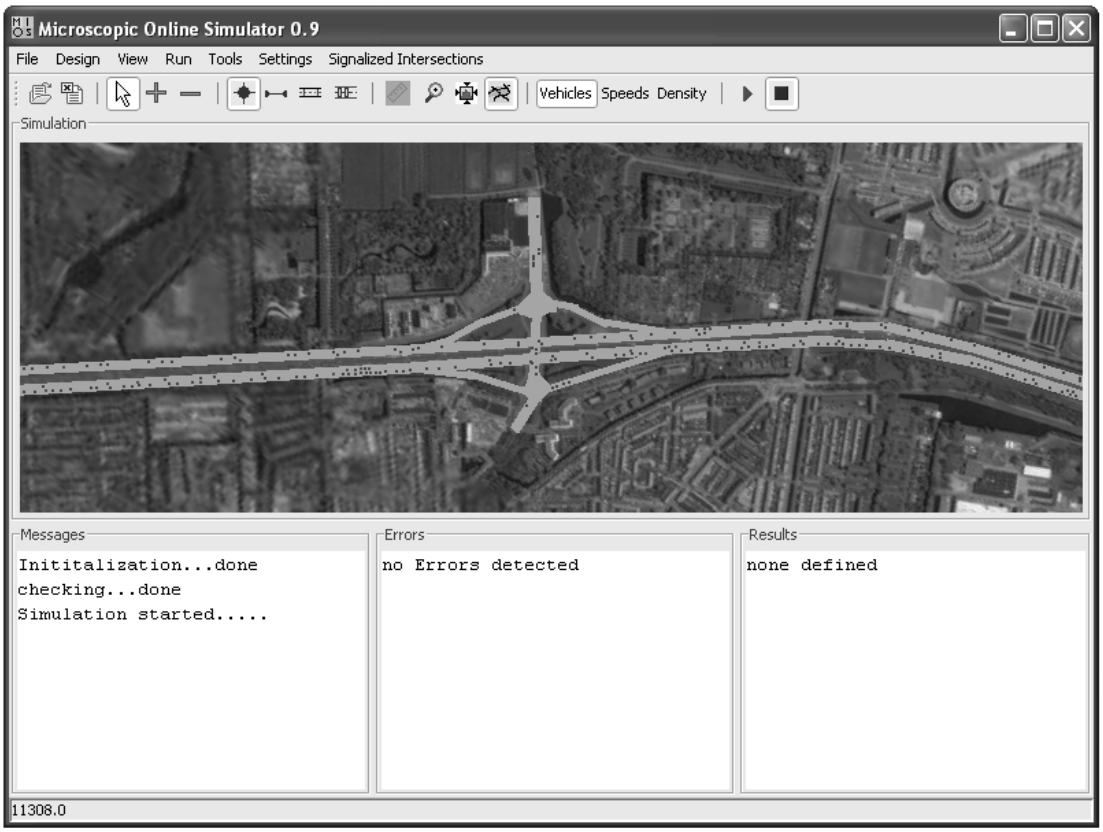

Figure 39: Simulation in MiOS showing individual vehicles

The cars are displayed in different colors according to the vehicle class. In the toolbar the visualization can be changed from “Vehicles” to “Speeds”, or “Density”, showing the average speeds and densities in a color scheme from green over yellow to red, indicating the values.
For further information during the simulation, the values for the placed detectors in the network are recorded every minute and displayed in a separate graph per detector (see Figure 40).

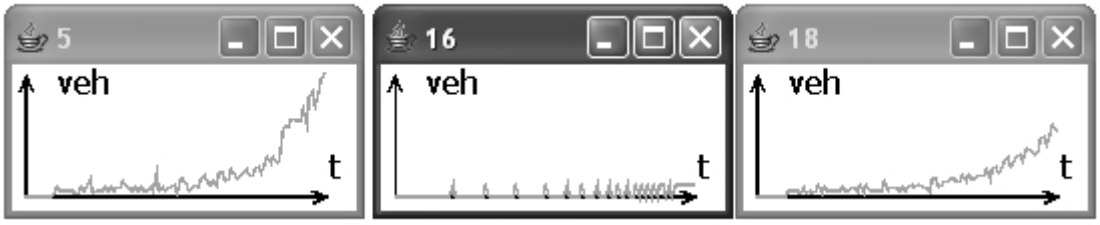

Figure 40: Detector values shown during runtime

In the following we describe the outputs of the simulation and the possibility to analyze them in a post process.

5.3 Post-processing of the simulation results

The user can define the outputs of the simulation that should be stored. These outputs can be:

- vehicle trajectories (position over time),
- travel times per OD pair (time elapsing from a vehicle to enter the network to its exit of the network),
- delays (time lost by driving with speeds below the desired speed, stored by link and network wide),
- stops (amount of times the vehicle stands still -> speed = 0 cells/time step), and
- logs of signal and controller settings (green times, VMS messages,…).

Outputs are stored in a database for further analysis. The time step of storing the data is limited by the capacity of the database, but since the necessary hardware gets less and less costly, this should not be a problem nowadays. The user defines next to the data that should be stored the place where to store it. To avoid redundancy of data MiOS works with linked databases. With different modules accessing the database(s)

used for storing the results a “software semaphore” is used to guarantee the integrity of the data. Semaphores are used in hardware design to overcome the possibility of deadlocks and data integrity in the memory. Since a semaphore is by definition a hardware method it is not correct to name it a semaphore, but since the software implementation does guarantee just single access to a table at a time the term “software semaphore” is introduced. In this way the data can be stored distributed over different locations of the computer network. In case of a system or transfer failure, MiOS sets up an emergency database to avoid the loss of any data and informs the user immediately.
To analyze the simulation results MiOS offers the possibility to create fact sheets as a report from the stored data in the database. Instead of using predefined tables and output files, MiOS allows the user to select certain parameters and aggregation time steps to create user specific files. To increase the portability of these files, the data size can be limited. This allows the export to common office application as Microsoft Excel®. Due to the concept of the MiOS framework, this can be done from every computer connected to MiOS. In this way the post-processing of the simulation results has no effect on the computation speed of the simulation.
An additional graphics interface is added to MiOS for an easy visualization. With this tool it is possible to create 2-dimensional graphs of selected values. Due to the implementation the amount of data points is just limited by the physical memory of the computer and not by the software itself, which allows visualizing more data at one time. The data can also just be exported to a data file that can be used in further applications like Matlab® or simple viewer tools like Surfer®.
This tool is not a supplement for a full statistical analysis of the simulation results, but is able to give a fast overview which is sufficient in many cases as a first step

5.4 Summary

This Chapter described the working steps with the combination of FrOnT and MiOS for running a microscopic online simulation. First the network elements and real-time data sources are defined and stored in the FrOnT database. Then it is translated to a MiOS network and gets simulated by using the MiOS simulation modules for OD estimation, route choice, driving behavior and of course the cellular automata.
The results of the simulation can be visualized directly in the FrOnT – MiOS application or can be exported to data files for further usage in other software packages like Matlab.
In Chapter 6 the application is tested in a real-world test case – the city of Delft. First the models performance is tested separately for a motorway and an urban road, before

running a simulation of a combined network of the whole city of Delft. The real-time measurements for these tests are from the Regiolab Delft (see Chapter 1).

6 Real world test case: “Delft”

To test the FrOnT – MiOS combined application, the region with the city of Delft as centre has been chosen. The Regiolab Delft offers real-time measurements from the surrounding motorways with double inductive loop detectors and several camera locations in the city itself. The first test case will focus on a motorway network after which the test is done on an urban road with major signalized intersections, and then finally to the combined network of Delft.
The tests are carried out with a combination of FrOnT and MiOS. FrOnT acquires the online measurement data via an Internet connection from the Regiolab Delft server to set up a starting point for the simulation. The network gets filled according to past flows and entry points with no measurements are set to a static OD matrix of the time interval that is simulated. The driving behavior model uses the training data gathered from remote sensing data of Dutch motorways and probe vehicle data from Japan. The simulation time step is due to the CA model set to 1/10s. Parameter settings of the route choice model are as follows:

$\alpha = 0.2$,

$\beta = 0.75$,

$T^{motorway} = 1.0$, $T^{urban} = 1.75$, and $T^{city} = 2.5$

After the network is filled with the initial set of vehicles the simulation runs for 30 minutes to warm up. During this period the control agents checks the differences between the simulation and measurements and the OD matrix tools calculates its parameters. After the warm-up phase, the simulation uses the real time information from the Regiolab Delft server to adjust the OD matrix and the simulation continues in real-time, estimating the actual traffic state. In case that simulation results and real-world data show a discrepancy of more than 10% in aggregated flows or speeds, the simulation is restarted from the last stored probable traffic state over to adjust the parameters for a better estimate.
Based on these estimates, a prediction is started to forecast a probable future traffic state. As a measure of success the travel time is chosen for the test on the motorway and urban road separately and vehicle volumes for the integrated test of the Delft network including route choice. The prediction of the traffic forecast is performed ten times faster than real-time.

6.1 Performance on motorway A13

6.1.1 Data input

The first test is done on a stretch of the motorway A13 in the west of Delft (see Figure 41). This part of the motorway is covered with double loop detectors, measuring flows and speeds in aggregated one minute values. The motorway is the major connection between The Hague and Rotterdam.

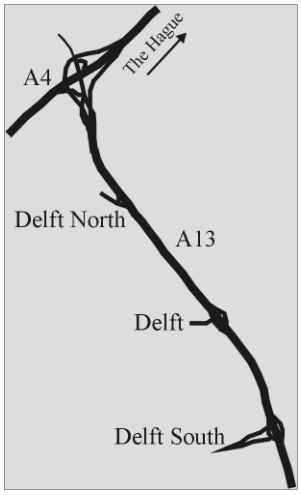

Figure 41: Map of the A13 motorway at the City of Delft

The network was created with the MiOS network editor and connected with the Regiolab Server for the real-time traffic measurements of speeds and flows at the network boundaries. The following four days have been selected for the performance tests:

- 20th of August 2003,
- 10th of December 2003,
- 24th of March 2004,
- 16th of June 2004,

which have significantly different patterns of the morning and evening peak. The Regiolab Server allows requesting data for a specific day and time of day. Even though the selected days are in the past, the connection to the Regiolab Server is the same as for requesting actual traffic measurement data.

Travel time is used as a performance measure and since the travel time for the motorway could not be observed directly, travel time estimation developed by Van Lint (2004) has been used for comparison with the prediction result from MiOS. Figure 42 shows the estimation results for the freeway A13 in different seasons, using the method from Van Lint for the stretch from the ramp Delft South to the ramp Delft North. The horizontal time axis marks the departure time from the point at the hight if the off-ramp Delft South.

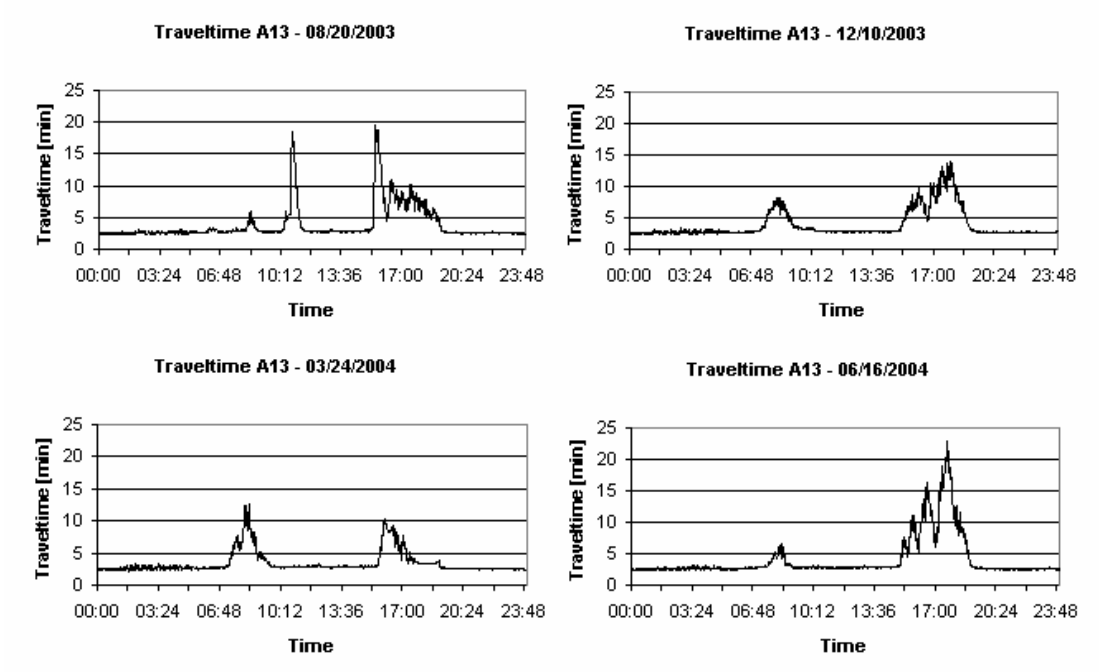

Figure 42: Estimation of daily travel times on the motorway A13 in different seasons

It can be seen, that there were big differences in the daily travel time patterns. The common characteristic was the free flow travel time of about 160 seconds. Significant are the peaks in the morning and in the evening with travel times up to 1380 seconds.

6.1.2 Travel time simulation of the A13

On the motorway we looked at four different scenarios. We picked parts of the morning peak in August, 20th 2003 and March, 24th 2004. These scenarios have different shapes. In August the peak consisted of a steep increase of travel time and a smooth decrease, while in June the travel time is more oscillating. Figure 43 shows the results of the morning peak in August. Between 10.00 am and 10.45 am the

predictions fit the measurement, but it takes longer to predict the steep increase. Due to the fact that the peak is well represented, the results are satisfying. The smooth decreasing travel time from 11.00 am on is well predicted.

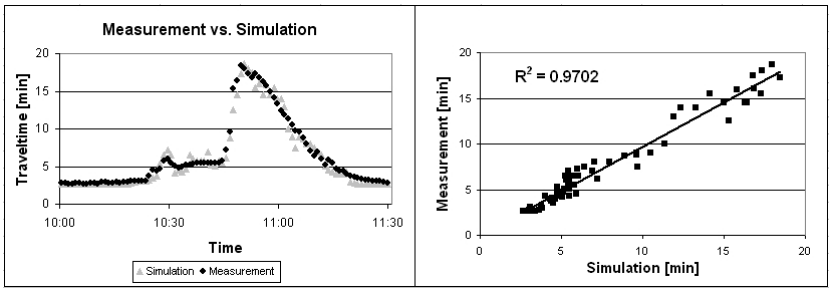

Figure 43: Travel time comparison of the morning peak in August

The results for March, presented in Figure 44, show that the oscillation around 8.00 am and 8.30 am leads to prediction errors of about 20%, but the peak is still represented well.

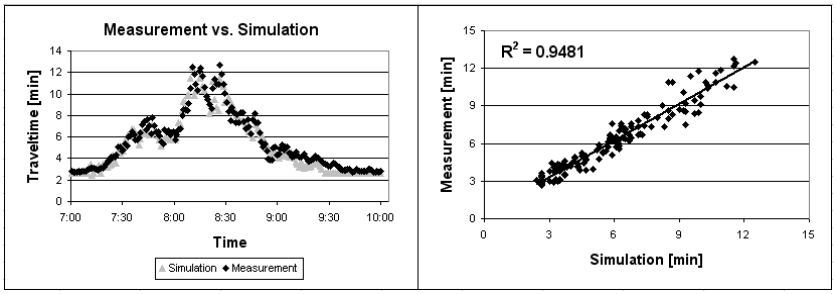

Figure 44: Travel time comparison of the morning peak in March

The evening peaks were chosen from a day in December, 10th 2003 and June, 16th 2004. Patterns of the evening peaks are less sharp than the morning peaks and we focused on a significant 2 hours period. The simulation of December shown in Figure 45 shows that the simulation results are changing more smoothly than the measurements, which leads to higher errors. The cause of this is based on the adjustments of the OD matrix in MiOS, which is a smooth process and in this case can not reflect the changes of demand in the real world. Nevertheless, the shape is well represented.

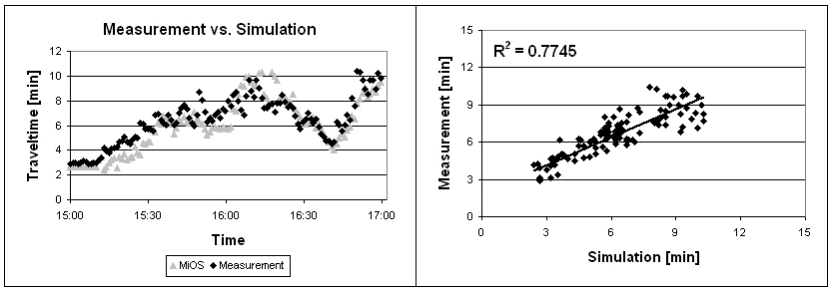

Figure 45: Travel time comparison of the evening peak in December

The measurements in June are quite smooth, and the simulation results, presented in Figure 46, show a smaller error in the predictions. It is evident, that the errors are in the same range, independent from the travel time. Also here, the biggest errors are found during oscillations in the measurements.

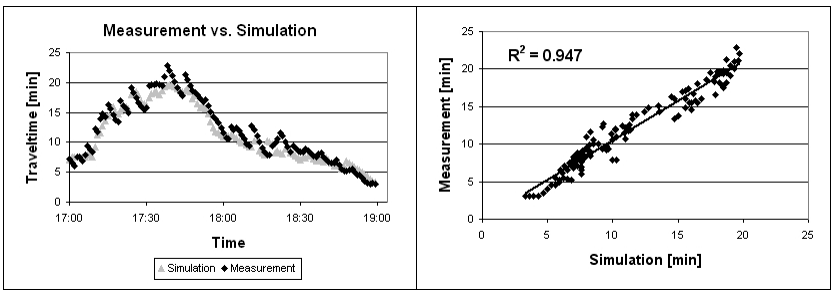

Figure 46: Travel time comparison of the evening peak in June

It can be seen clearly, that a steep increase or decrease of travel times leads to greater differences between the prediction and the measurement. This is caused by the reaction time of the simulation model, which adjusted the parameters since the threshold for a reset were not given at any point. The simulation covers the maximum values of the travel time, so that the delay should not have a great impact on applications, which are based on the predictions.

Dataset	RMSE	Bias	RRE	MAPE
August	0.822 min	-0.204 min	0.797 min	10.66 %
March	0.651 min	-0.246 min	0.603 min	5.20 %
December	1.118 min	-0.307 min	1.046 min	12.00 %
June	1.340 min	-0.569 min	1.213 min	8.94 %

Table 3: Statistical comparison for the August data set

The statistical comparison of the measurement data and the simulation results is done by calculating the root mean square error (RMSE), the bias for structural errors and the root residual error (RRE) for the random errors. Additionally the mean absolute percentage error (MAPE) is calculated where the negative and positive errors are not balancing out each other. The values are summarized in Table 3.
In a next test we used camera detection data which provides individual travel time data during the day. The performance of MiOS is shown in Figure 47.

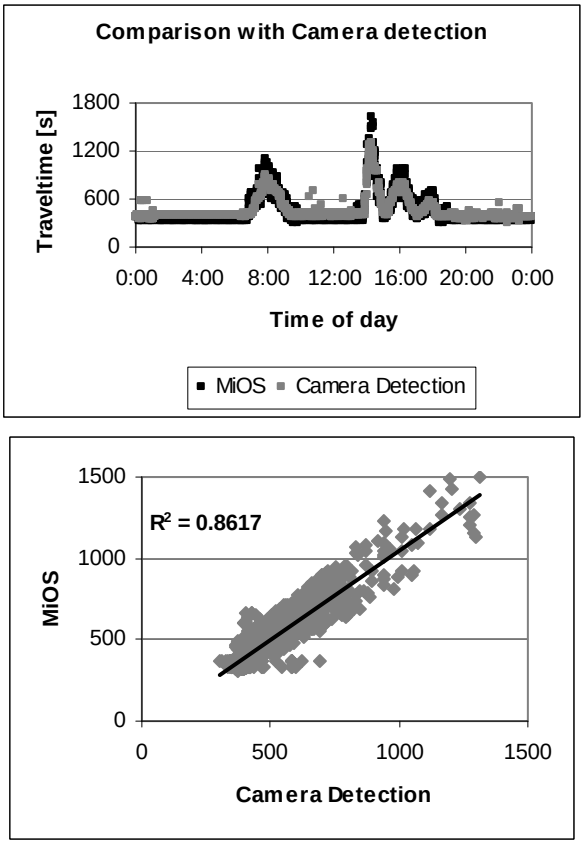

Figure 47: Comparison of MiOS and camera detection data on the A13

6.2 Performance on urban road N470

6.2.1 Data input

The urban road N470 in the South of Delft connects the motorway A4 with the motorway A13 and therefore a vital road. The Regiolab Delft has four cameras on this road combined with number plate recognition, so that individual travel times can be

measured. The N470 and the camera locations along the road can be found in Figure 48.

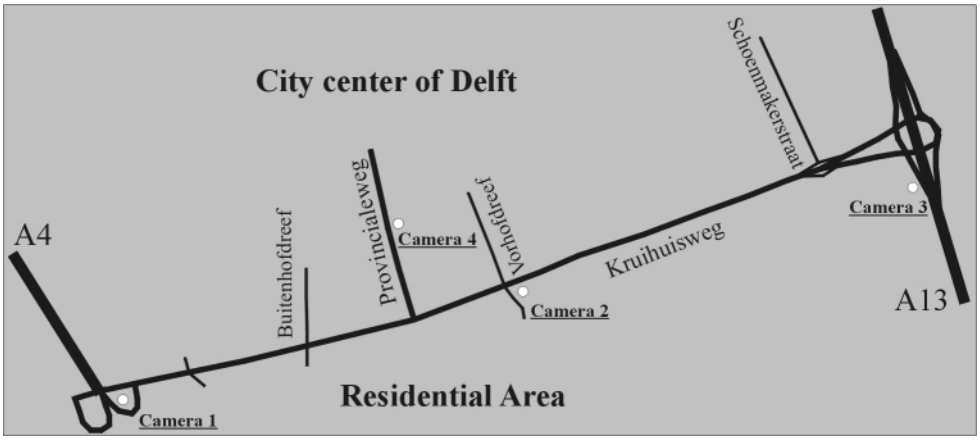

Figure 48: Map of the urban road N470 in the South of Delft

Cameras 1 to 3 are just working in direction west-east. To have an indication for the flow in the opposite direction the counting loop detectors at the intersections of the N470 are taken into account. A last characteristic of this road is a drawbridge between the Voorhofdreef and the Schoenmakerstraat which is operated during the day and can induce severe traffic congestions. The operation of the drawbridge is not fixed scheduled and no information is online available.

6.2.2 Simulation of the N470

As mentioned before, the focus during the simulation is on the vehicle coming from the A4 in the West going to the A13 in the East of Delft. The performance measure is again the travel time. Cars passing camera 1 get registered and when passing camera 3, the individual travel time is calculated.

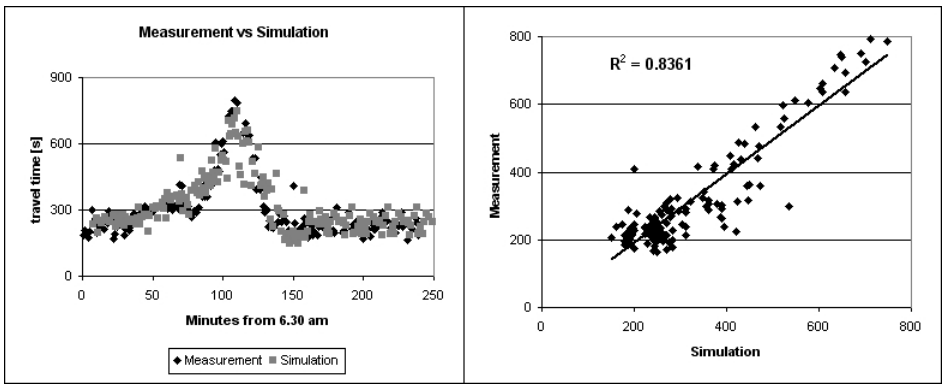

Figure 49: Comparison at the urban road N470

These individual travel times are then compared with the simulation results. Figure 49 shows the comparison of the one minute aggregated values and their regression. During the peak (between 7:45am and 8:35am) the travel time gets underestimated, but shows the clear trend. Further statistical analysis resolved in a RMSE value of 77.124 s, split in a bias of -18.24 s and a RRE of 74.936 s. The mean absolute percentage error for the simulation was 12.34 %.

6.3 Integrated testing on the Delft network

6.3.1 Data input

After testing part of the FrOnT – MiOS application on a motorway and an urban network separately, the following simulation is done for an integrated network of the City of Delft. It contains the surrounding motorways A4 and A13, the urban connections of the motorway N470 and the major inner-city roads. Therefore, route choice is included into this test. The focus is on the vehicles traveling from the South (Rotterdam), going to The Hague in the North. As input the online data from the 25th of February 2004 is used. On that day an accident on the A13 happened and the motorway was highly congested. According to this, the amount of traffic on the N470 in the south of Delft between the two motorways was much higher.
The online simulation run was started with the recorded data from one hour before the incident and after the detection of the accident the first prediction, using the described route choice model.

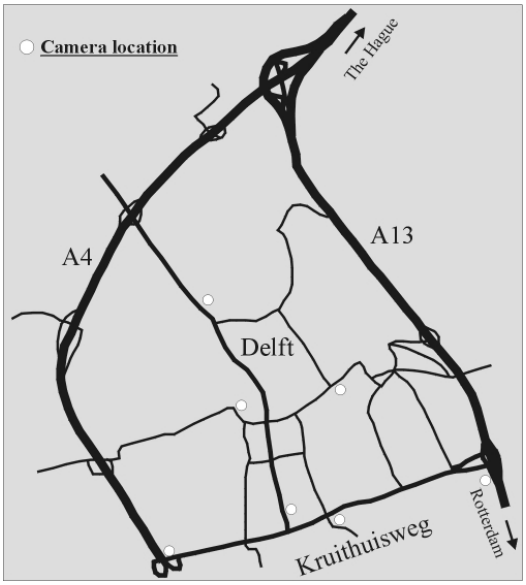

Figure 50: Map of the City of Delft

The measurements of this day are shown in Figure 51 compared to the values of one week before. It shows that around 10.50h an accident happened on the A13 direction Rotterdam to The Hague between the off-ramp Delft South and the off-ramp Delft.

After the A13 was completely blocked the traffic on the A4 starts increasing to about the double amount of a usual day by the demand from the city of Delft. At about 14.30h the closure of the A13 was over and it took about one hour until the normal traffic conditions on both roads were restored.

Another interesting part of the measurement is the peak around 12.00h. It is assumed that the police started to reroute cars to Rotterdam over the off-ramp Delft to clear up the situation. But here the interest is in the shift of traffic going through the city and vehicles using the N470 in the South of the city. The test aims to investigate if the model can predict the increasing demand of the A4 during the time of the incident.

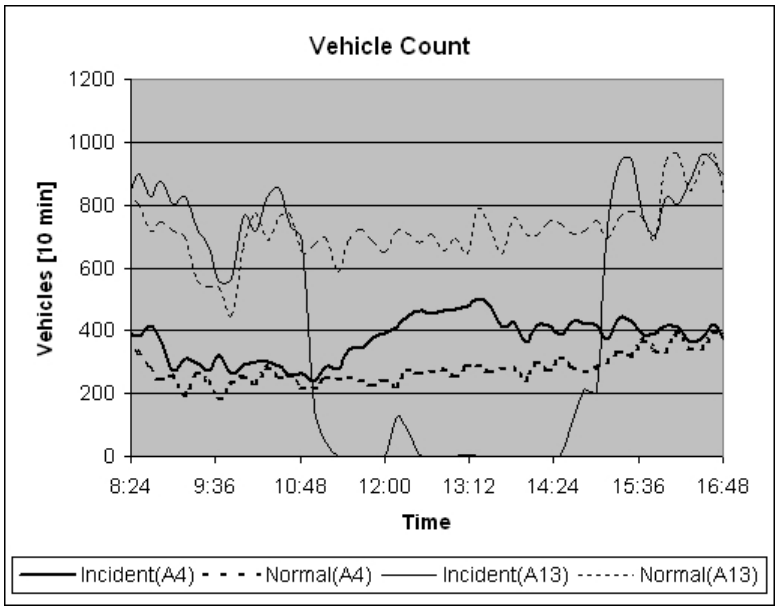

Figure 51: Measurements during the incident compared to normal conditions

6.3.2 Simulation of the Delft network

First we have simulated the time interval from 9.40h until 15.40h under normal traffic conditions, to calibrate the MiOS model. The bases for the normal traffic conditions are the online measurements from the day before the incident.

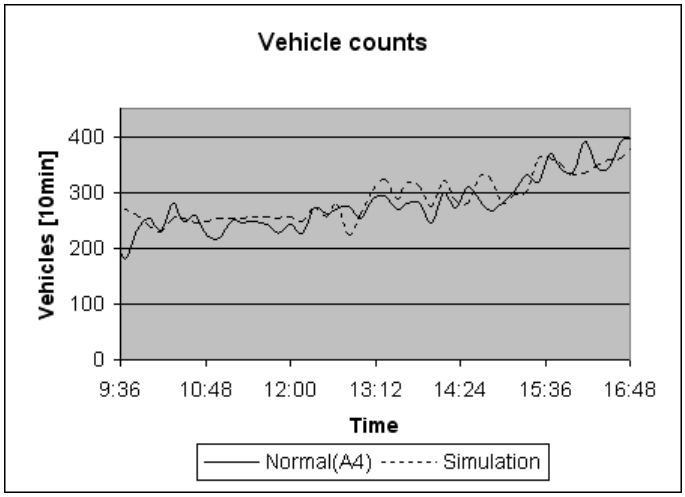

Figure 52: Comparison on the A4 under normal conditions

The results are shown in Figure 52 and Figure 53. Even if the model smoothes the actual counts, the outcome follows the actual trend. For both motorways the beginning evening peak is well represented. Due to the fact, that under normal conditions the A4 is not an alternative the results of both motorways are nearly independent and the route choice model does not affect the traffic counts for the traffic using the A13 from the South to the North.

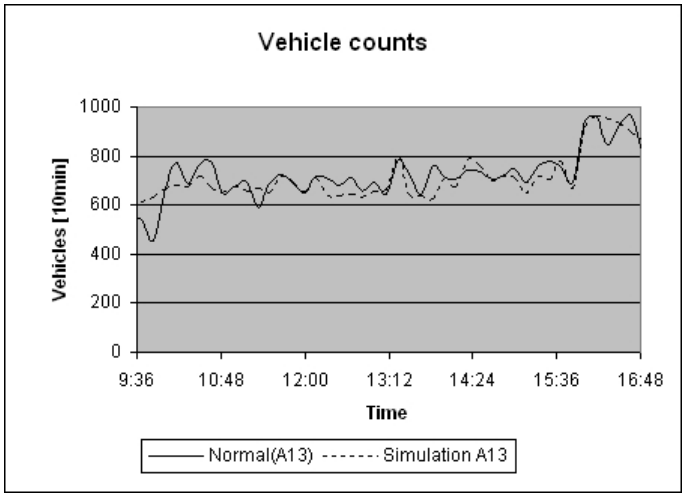

Figure 53: Comparison on the A13 under normal conditions

For the simulation of the incident we have focused on the motorway A4, because the A13 was blocked and the results would not be significant. While the real-time simulation estimates the actual state, the prediction makes forecast 30 minutes ahead every 5 minutes. Figure 54 shows that the simulation predicts the shift to the

motorway A4. With this information traffic engineers could adapt the traffic control strategies on the N470 to cope better with the higher demand on that road.

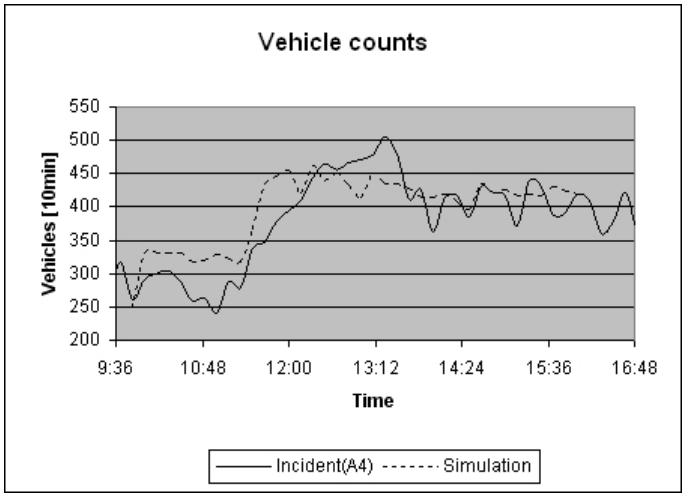

Figure 54: Comparison on the A4 during the incident

6.4 Conclusions

In this Chapter we have shown that the FrOnT – MiOS combination is able to predict the travel times on motorways and urban arterial roads as well as its capability to show the effect of incidents in terms of demand distribution. It is therewith proven, that the combination of autonomous modules is able to simulate and to predict traffic network conditions and that the design of the system is able to handle the tasks of estimation and predictions.

Monitoring the computational performance we found that running FrOnT and MiOS on a single computer allows the handling of up to 8500 vehicles with the time step of 1/10s for a real-time estimation and predictions 30 minutes ahead every 5 minutes. The Delft network caused no critical computational load for the route choice and the data gathering in one minute intervals performed without data loss or time-outs from neither server-side nor FrOnT-side.

Due to the limitations of the single modules of MiOS we refrained from further tests, since these would not be able to indicate flaws in the system architecture and just illustrate the limitations of the prototype simulation modules, which are basic implementations of ideas along the research performed.

In the following Chapter we draw the conclusions found from this work and give suggestions for further research regarding FrOnT, MiOS and processing of traffic data.

7 Conclusions and further research

In this thesis we have introduced a framework for online simulation that allows users to work with a single system containing all necessary elements for simulation, data management, data processing, and communication, and to integrate state of the art tools for the single tasks of the process of traffic management without changing the framework. This framework, coupled with simulation tools was then tested in the Regiolab Delft area, using online traffic measurement data as input to the model.
Several new model components have been added such as a new driver behavior model based on believe networks and a route choice model based on travel costs, travel information and comfort. A simple existing model for OD estimation has been implemented.
We have shown that the FrOnT framework can incorporate independent modules to build a system to support traffic management, which enables researcher and practitioners to build their own support tool in a very flexible way. By distributing the tasks in autonomous modules which communicate with each other via the FrOnT framework, the computational load can be spread.

7.1 Conclusions per Chapter

Since the work contains two major parts and several aspects of traffic engineering and software design, the following section will give the conclusions of this thesis separate for the different Chapters for a better structure.

- **Chapter 1:**

  The requirements for practitioners for a traffic control and information system are based on the daily use. They prefer a single system with one user interface over several systems they have to use and maintain. For researchers it is interesting to have a system to test their new models and control systems within a consistent environment with synthetic data as well as real world measurements.
  So there is a need for a software tool that satisfies the simplicity to work with, with the complexity of handling specialized software modules for the best possible results for state of the art research.

- **Chapter 2:**

  Inputs for all tools of online traffic management are measurements from the network. These inputs come from various equipment and systems and needs to be generalized in a way that the further handling of the data can be independent from its source. Therefore, we presented a suitable database design for the storage of roadside measurement data, which allows the integration of data collection to the system. This structure is open to make it possible to add new kinds of measurements.

  In the following we designed a multi agent framework to process the data. Several agents, combined to an agent society, allow to separate the tasks of data collection, data processing (simulation, analysis …) and communication and to synchronize them throughout the system. This design allows researchers to test new algorithms by plugging their own algorithm into the system. Instead of knowledge about the FrOnT system itself, or any other used module of the framework, the researcher just has to fulfill simple interface requirements by following a protocol for the data exchange with the framework. This saves research time for developing test-beds for new tools and gives the opportunity to compare different approaches in a stable environment.

  FrOnT is closing the gap between the specialized research (OD-estimation, route choice, behavior modeling …) and the software development which is used in practice. It accelerates the usage of state of the art technology in traffic control centers as well as in research facilities without the delay of reengineering the existing system. With its implementation in the platform independent programming language JAVA it can run on several environments and can be distributed in even heterogeneous computer networks consisting of personal computers and workstations.

- **Chapter 3:**

  Online simulation tools on the market and further developments are mostly based on former offline versions. The difference between the online and offline versions is the possibility to include online data as

an input to the simulator. OD estimation tools are added or serve as another input. E.g. DYNASMART-X is redesigned as an online version and converted to a CORBA based distributed system. All systems are designed as a suite of tools working together and offer application interfaces for interaction, but do not allow a simple change of certain algorithm and model parts, which is part of the commercialization of the products. The simplicity of FrOnT and the possibilities of exchanging software modules without further knowledge on the system are new and simplify the testing of new algorithms in a real-time environment.

- **Chapter 4:**

  To test the abilities of the FrOnT framework, we developed the necessary basic modules for a microscopic online simulation: Simulator, route choice model, driving behavior model, and OD estimator.

  Simulator
  For the microscopic simulation we chose a structure based on a cellular automaton for fast implementation. By decreasing the cell length of the automaton we extended the basic CA model for traffic flow by the possibility of simulating heterogeneous traffic and to give a better representation of distances, gaps and speeds. The resulting update time decreased therefore to 0.1s. Since the goal was to distribute the different parts of simulation into autonomous modules, the updating rules are exchanged by a behavior model which we describe in the following.

  Driving behavior model
  Inspired by the new microscopic measurement data that recently is becoming available; we decided to introduce a new driving behavior model based on this data. The psychology behind the driving task is very complex and the existing behavior models are not able to represent the real world. So we used the real world data to train a Bayesian network. A pattern matching between the status of a driver and his reaction set defines the behavior of a driver in the simulation

model. With more data available in different situations and locations, the developed driving behavior model can learn how drivers react to certain situations. The status of a driver as an input to the model is given by the simulator. Even though this model needs further refinements by gathering more distinct observations to build up training sets, it showed in the tests that it is leading to plausible simulation results. Further work is needed to validate the model in isolated situations of weaving section and other critical spots for microscopic simulation models.

Route choice model

The calculation of the route between the origin and destination is done by a new route choice modeling approach based on the actual travel time in the network and vehicles can update their route on their way. Next to a standard impedance function to calculate the costs of a link based on road type, and travel time, the model also incorporates en-route information and route guidance equipment. The addition of an individual threshold of comfort that determines the timing of a rerouting is new and gives a further tuning parameter to model the composition of individual drivers.

OD estimator

The OD estimator is a basic module which multiplies OD pairs with a correction factor according to the real world measurements. It is very basic and just for testing purposes.

- **Chapter 5:**

The implementation of the user interface demonstrated that the framework and the simulation can be controlled from an easy to use application interface. The complexity of the system and the synchronization of all involved modules remain hidden, which makes the system user friendly. Different interfaces for different user groups could be generated to control the possible options for administrators, and traffic engineers.

- **Chapter 6:**

  The basic test and validation of the framework and its modules was performed on the road network around the city of Delft. We have shown that the system is able to predict travel times on the motorway A13 as well as the urban road N470. A test on the combined network of the city of Delft showed that in case of an incident the volumes on the observed roads are shifting according to the measurements.

  These satisfying results show that the connection of the FrOnT framework and the MiOS modules for a microscopic online simulation is suitable for online traffic management and could serve as a support for traffic engineers in control centers to optimize the network operations.

7.2 Further research

In this section we present topics for further research. The topics are grouped into the categories *FrOnT*, *MiOS*, and *traffic data*. They refer to information technology as well as traffic modeling.

FrOnT:
The framework is programmed as a prototype and offers possibility for improvements in memory management and messaging optimization. For maintenance of the system a web based management tool could be added, to allow a remote configuration through the internet. Further extensions could be an automatic detection of computation loads and automatic balancing of computation task among the framework.

MiOS:
The modules of MiOS are basic modules which can be improved. The driving behavior model can be trained with additional measurement data and comparisons of behavior in different countries could be performed. Further it would be interesting to study the influence of weather and road conditions on a microscopic level.
To increase the speed of the simulation a hybrid simulation with macroscopic and microscopic simulators can be performed to study the gain and losses of one or the other approach. For the route choice it would be interesting to use different

impendence functions and to incorporate results from travel behavior research with given en-route information.

Traffic data:

Gathering real life data is a time consuming job in research. A lot of data is measured and stored in several places and different formats. While a lot of it is not used, other institutions gather similar data on different locations. In this way a lot of money and time is spend unnecessarily. The developed FrOnT framework with its structure of data storage can be extended by a web based data source to provide researchers with an easy access to the data they need in a format they can use. With the FrOnT data structure used in databases, a second layer for data fusion and conversion could filter the applicable measurement data sets and deliver them to the end user in a predefined format.

8 References

Abrahamsson, T., 1998, Estimation of Origin-Destination matrices using traffic counts – a literature survey, Report no. IR-98-021, International Institute for Applied System Analysis

Algers, S., Bernauer, E., Boero, M., Breheret, L., di Taranto, C., Dougherty, M., Fox, K., and Gabard, J. (1997) Review of micro-simulation models. Smartest Project deliverable D3. Leeds

Barceló, J. et al, 1996, 'PETRI: A parallel environment for a real-time traffic management and information system. In: Proceedings of the 3rd World Congress on ITS (published on CD-ROM), Orlando, Florida

Barceló, J., Casas, J., Ferrer, J.L. & García, D., 1999, "Modeling Advanced Transport Telematic Applications with Microscopic Simulators: The case of AIMSUN2", in: *Traffic and Mobility, Simulation, Economics, Environment*, W. Brilon, F. Huber, M. Schreckenberg and H. Wallentowitz (Eds.), Springer.

Bell, M.G.H. et al, 1997, 'A stochastic user equilibrium path flow estimator', *Transportation Research Part C: Emerging Technology*, Volume 5, Issues 3-4, 10 August 1997, Pages 197-210

Bell, M.G.H., 1991, The real-time estimation of origin-destination flows in the presence of platoon dispersion, *Transportation Research 25B*, 115-125

Ben-Akiva, M.; Bierlaire, M.; Burton, D.; Koutsopoulos, H. and Mishalani, R. (2000). Simulated-based tools for dynamic traffic assignment: DynaMIT and applications. Proceedings of the ITS America 10th annual Meeting May, 01-04, 2000

Boer, E.R, and Hoedemaeker, M., 1998, Modeling driver behavior with different degrees of automation: A hierarchical framework of interacting mental models. In Proceedings of the 17th European Annual Conference on Human Decision Making and Manual Control, Valenciennes, France

Bogers, E.A.I., H.J. van Zuylen, S.P. Hoogendoorn and K.A. Brookhuis, 2006, 'Learning in day-to-day route choice', paper presented at the 11th International Conference on Travel Behaviour Research, Kyoto, August 2006

Brackstone, M., McDonald, M. & Wu, J., 1997, 'Development of a fuzzy logic based microscopic motorway simulation model', In: Proceedings of the IEEE Conference of Intelligent Transportation Systems (ITSC97), Boston, USA

Buntine, W., 1996, 'A guide to the literature on learning probabilistic networks from data', *IEEE Trans. On Knowledge and Data Engineering*, Vol. 8, Berkeley

Cascetta, E., 1984, Estimation of trip matrices from traffic counts and survey data: a generalized least square estimator, *Transportation Research 8B*, 289-299

Cascetta, E., 2001, Transportation systems engineering: theory and methods, Kluwer Academic Publishers

Cascetta, E., Inaudi, D. & Marquis, G., 1993, 'Dynamic estimators of Origin-Destination matrices using traffic counts', *Transportation Science*

Cascetta, E., Nguyen, S., 1988, A unified framework for estimating or updating origin/destination matrices from traffic counts, *Transportation Research 22B*, 437-455

Cetin et. al, 2003, 'A Large-Scale Agent-Based Traffic Microsimulation Based On Queue Model', Swiss Transport Research Conference, Ascona

Chen, Peter P., 1976. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems 1* (1): 9-36.

Cheng, J. et al, 2002, 'Learning Belief Networks from Data: An Information Theory Approach', *The Artificial Intelligence Journal*, 43-90

Chrobok, R., Pottmeier, A., Marinosson, S. & Schreckenberg, M., 2002, 'Online Simulation and Traffic Forecast: Applications and Result', Gerhard Mercator University

Codd, E.F., 1970, "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM 13* (6): 377–387

Cox, B. J. 1986, *Object-oriented programming : an evolutionary approach*, Reading, Mass. : Addison-Wesley.

CROW, 2002, 'Guidelines for Highway Constructions' (Handboek Wegontwerp – Basiscriteria 164a)

Crutchfield, J.P., K. Kaneko, 1987, Phenomenology of spatiotemporal chaos, in: B.L. Hao (Ed.), *Directions in Chaos,* World Scientific, Singapore, pp. 272–353

Delorme, M., 1989, An introduction to cellular automata, in: M. Delorme, J. Mazoyer (Eds.), *CellularAutomata—a Parallel Model*, KluwerAcademic Publishers Group

Dijker et. al., 2005, FOSIM User manual, www.fosim.nl

Dijker, T., 1997, Verkeersafwikkeling bij congestie. Graduation Thesis (in Dutch), Transportation and Traffic Engineering Section, Delft University of Technology

Doniec, A. et al., 2005, Dealing with multi agent coordination by anticipation: Application to the traffic Simulation at Junctions.

Engberg, A. ,2001, 'Notes on Integration of the Paramics Simulation Software with the ATON System' ATON report

Espié, S. et al., 1994, Microscopic traffic simulation and driver behavior modeling: The ARCHISIM project, INRETS

Fernandes, C.M., 2004, 'An Algorithm to Handle Structural Uncertainties in Learning Bayesian Network', Brasil

Fildes, B.N. et al, 1989, 'Speed Perception 2: Drivers' judgments of safety and speed on rural straight and curved roads and for different following distances', Report of the Australian Transport Safety Bureau

Florian, M., Chen, Y., 1991, A bilevel approach to estimating OD matrix by traffic counts, CRT research report no C7PQMR PO750-X

Foster, I., 1995, *Designing and Building Parallel Programs*, Addison-Wesley, Reading, MA

Gipps, P.G., 1981, 'A behavioural car following model for computer simulation', *Transportation Research B*, 15, 105-111

Gipps, P.G., 1986, 'A model for structure of Lane changing descicions', *Transportation Research 20B*, 107-120

GMD Ais, 2000, 'City Traffic – Integriertes System fuer Verkehrsplanung, -Management und –Information in urbanen Ballungsraeumen, GMD Forschungszentrum, Instut fuer Autonome Intelligente Systeme, Sankt Augustin, Germany

Grol, H.J.M. van, 1997, 'Online Network State Estimation and Short Term Prediction', DACCORD project deliverable D05.2

Haas, C.P., 2001, 'Assessing Developments using AIMSUN', Institute of Professional Engineers New Zealand

Hazelton, M.L., 2000, Estimation of origin destination matrices from link flows on uncongested networks, *Transportation Research 34B*, 549-566

Heckerman, D., 'A tutorial on learning with Bayesian networks, Learning in graphical models', MIT Press, Cambridge, MA, 1999

Heusch&Boesfeldt GmbH, 2004, 'Software Architecture of a Traffic Control and Information Center', www.heuboe.de

Hidas, P., 2002, 'Modelling lane changing and merging in microscopic traffic simulation', *Transportation Research Part C 10*, 351-371

Hoogendoorn, S.P., P.H.L. Bovy and H. van Lint (2002). Short-term prediction of Traffic Flow Conditions in a Multilane Multi-class Network. In Michael A.P. Taylor, (Ed.), *Transportation and traffic theory in the 21st century*. (pp. 625-651). Oxford: Pergamon, Elsevier Science

Hoogendoorn, S.P., S.J.L. Ossen, 2005, Parameter Estimation of Car-Following Models. 16th International Symposium on Transportation and Traffic Theory, 245-266

Hoogendoorn, SP, & Botma, H, 1997, Modeling and estimation of headway distributions. *Transportation research record*, 1591, 14-22

Hoogendoorn, SP, & Zuylen, HJ van, 2004, Tracing Congestion Dynamics with Remote Sensing. In C Brebbia (Ed.), *Management and Information Systems incorporating GIS and Remote Sensing* (pp. 309-319). Southhampton: WIT Press

Hoogendoorn, SP, Schuurman, H, & Schutter, BHK ,2003, Real-time traffic management scenario evaluation. In S Tsugawa & M Aoki (Eds.), Proceedings of the 10th IFAC symposium on control in transportation systems (pp. 343-348). Tokyo: IFAC / IFORS

Horvitz, E. J., 1987, Problem-solving design: Reasoning about computational value, trade-offs, and resources. In Proceedings of the Second Annual NASA Research Forum, pages 26--43, Moffett Field, California, NASA Ames Research Center

Huang, T., D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell and J. Weber, 1994, Automatic Symbolic Traffic Scene Analysis Using Belief Networks. Proceedings of the Twelfth Conference on Artificial Intelligence

Hunt, J.G. & Lyons, G.D., 1994, 'Modelling a dual-carriageway lane changing using neural networks', *Transportation Research Part C* 2 (4), 231-245

Jorgensen, H., 2002, Looked but failed to see errors in traffic, *Accident Analysis and Prevention*

Kaneko, K., 1990, Simulating physics with coupledmap lattices, in: K. Kawasaki, A. Onuki, M. Suzuki (Eds.), *Formation, Dynamics, and Statistics of Patterns*, WorldScientific, Singapore, pp. 1–52

Kjaerulff, U., 1992, Reduction of computation complexity in bayesian networks through removal of weak dependencies. In Proc. of UAI-94, pages 374—382

Kremer, M. et al., 1987, A new class of dynamic methods for identification of origin destination flows, *Transportation Research 21B*, 117, 121

Lindveld, Ch.D.R., 2003, Dynamic O-D matrix estimation: a behavioural approach , T2003/7, September 2003, TRAIL Thesis Series, Eburon, The Netherlands

Lint, H. van, 2004, 'Reliable Travel Time Prediction for Freeways', TRAIL thesis series, Netherlands

M. Treiber, A. Kesting, and D. Helbing, 2005, 'Delays, inaccuracies, and anticipation in microscopic traffic models', *Physica A 360*, 71-88

Maerivoet, S., et.al., 2005, Cellular automata models of road traffic, *Physics Reports* 419, 1-64, Elsevier

Mahmassani, H., 2000, Trip timing, Handbook of transportation modeling, eds, Hensher, D.A. and Button, K.J. Pergamon

Mahmassani, H., Qin, Xiao and Zhou, Xuesong, 2004, DYNASMART-X Evaluation for Real-Time TMC Application: Irvine Test Bed, Maryland Transportation Initiative, University of Maryland, College Park, Maryland

Mahmassani, H.S., and Zhou, X., 2005 "Transportation System Intelligence: Performance Measurement and Real-Time Traffic Estimation and Prediction in a Day-to-Day Learning Framework," Chapter 16 in Advances in Control, Communication Networks, and Transportation Systems, In Honor of Pravin Varaiya, edited by E. Abed, Birkhauser.

Mahut, M. et al., 2002, 'Application of a simulation-based dynamic assignment model', IEEE 5th ITS conference, Singapur

Mahut, M. et al., 2003, 'Space-Time Queues and Dynamic Traffic Assignment: A model, algorithm and applications', Transportation Research Board, Washington DC

Mahut, M., 2001, 'A multi-lane extension of the Space-Time Queue Model of Traffic Dynamics', TRISTAN IV, Azores Islands

McDonald, M., Brackstone, M. & Jeffery, D., 1994, 'Simulation of lane usage characteristics on 3 lane motorways', In: Proceedings of the 27th ISATA Conference, Aachen, Germany

Messmer, Papageorgiou, 1990, A. Messmer and M. Papageorgiou, METANET: A macroscopic simulation program for motorway networks, Traffic Engineering + Control (1990), 466-470

Microsoft Support Technology, 1999, Microsoft Bayesian Networks, Basics of Knowledge Engineering, Microsoft, Redmont

Minderhoud, M.M.,1999, Supported Driving: Impacts on Motorway Traffic Flow, T99/4, July 1999, TRAIL Thesis Series, Delft University Press, Delft, The Netherlands

Miska, M., Muller, Th. & van Zuylen, H.J., 2004, 'MiOS – A Microscopic Online Simulator', In: CD-ROM Proceedings of the ITS World Conference 2004, Nagoya, Japan

Muller, ThHJ, & Zuylen, HJ van, 2002, 'Regiolab Delft', In 9th World Congress on Intelligent Transport Systems, Chicago, 14-17 oktober 2002 (pp. 1-9). s.l.: Mira Digital Publishing.

Nagel, K., 1996, Particle hopping models and traffic flow theory, *Phys. Rev. E* 53 (5), 4655–4672

Nagel, K., M. Schreckenberg, 1992, A cellular automaton model for freeway traffic, J. Phys. I France 2 2221–2229

Neumann,J. von, 1948, The general andlogical theory of automata, in: L.A. Jeffress (Ed.), *Cerebral Mechanisms in Behavior*, Wiley, New York, 1948, pp. 1–41, paper presentedat the Hixon Symposium

Ngoduy, D., 2006, Macroscopic Discontinuity Modeling for Multiclass Multilane Traffic Flow Operations, T2006/3, April 2006, TRAIL Research School, The Netherlands

Nielsen, P., Lisse, S. and Beard, J., 2003, Causal Models and Learning for Robust Human Behavior Models. Proceedings of the '03 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL

Oestereich, B, 2002, 'Developing Software with UML: Object-Oriented Analysis and Design in Practice', Addison-Wesley Professional; 2 edition

Ossen, S., Hoogendoorn, S.P. & Gorte, B., 2006, 'Inter-Driver Differences in Car-Following: a Vehicle Trajectory based Study', Proceedings of the Transportaion Reasearch Board Annual Meeting (TRB)

Pahl, P.J. & Damrath, R., 2001, *Mathematical Foundations of Computational Engineering*, Springer

Pearl, J., 2000, 'Causality: Models, Reasoning and Inference', Cambridge University Press, Los Angeles

Postans, R. L., & Wilson, W. T., 1983, Close-following on the motorway. *Ergonomics*, 26, 317–327

Quadstone Group, 2004, 'Paramics Online', Scottland

R. Balakrishna (2006) "Off-Line Calibration of Dynamic Traffic Assignment Models." PhD thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology.

Raney, B. et al., 2003, 'An agent-based microsimulation model of Swiss travel: First Results', *Networks and Spatial Economics* (3), 23-41

Riemersma, 1979, Perception in Traffic, *Urban Ecology* Vol. 4

Rothengatter, G., 1998, Traffic Psychology and Behavior, *Transportation Research Part F*

Rumelhart, D., Geoffrey Hinton, and Ronald Williams. 1986. Learning internal representations by error propagation. In Rumelhart and McClelland, eds, *Parallel Distributed Processing*, vol 1. MIT Press

Russell, S. et al, 1997, Batmobile: Towards a bayesian automated taxi. In Proceedings of the 14th Int. Joint Conf. on Artificial Intelligence, pp. 1878—1885

Searle, J, 1969, Speech Acts: An essay in the Philosophy of language, Cambridge University Press; New Ed edition

Sherali, H.D. et al, 2001, Estimation of dynamic origin destination trip tables for a general network, *Transportation Research* 35B, 217-235

Spirtes, P. et al., 2000, 'Causation, Prediction and Search', 2nd edition, The MIT Press, Cambridge

Sukthankar, R., 1997, Situation Awareness for Tactical Driving, PhD Dissertation, Carnegie-Mellon University

Tampère, C. M. J, 2004, Human-Kinetic Multiclass Traffic Flow Theory and Modelling (With Application to Advanced Driver Assistance Systems in Congestion). Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands

Toledo, T., 2003, Integrated Driving Behavior Modeling, PhD Dissertation, MIT

Torday A., Bert E. and Dumont A.-G. (2004) Route choice relevance in complex urban network micro-simulation models, proceedings of the 4th Swiss Transport Research Conference, Monte Verita

Troutbeck, R.J., Kako, S., 1997, 'Limited priority merge at unsignalised intersections', In: Kyte, M. (Ed.), Proceedings of the Third International Symposium on Intersections Without Traffic Signals University of Idaho, Moscow, Idaho, USA, pp. 294-302

University of Maryland, 2003, 'DYNAMSMART-X – User's Guide'

Van der Zijpp, N.J., 1996, Dynamic Origin Destination matrix estimation on motorway networks, PhD Thesis, Delft University of Technology

Van Winsum, W., 2000, Measuring drowsiness and impairment in car driving. In: Proceedings Vision in Vehicles 8, Boston, Augustus 1999

Van Zuylen, H.J., 1983, Some improvements in the estimation of an OD matrix from traffic counts. Proceedings of the 8th International Symposium on Transportation and Traffic Theory, Toronto Press

Viti, F, Bogers, EAI, & Hoogendoorn, SP, 2005, 'Day-to-day learning under uncertainty and with information provision: model and data analysis', In HS Mahmassani (Ed.), Proceedings of the 16th International Symposium of Transportation and Traffic Theory (pp. 1-21). Maryland: University of Maryland. (TUD)

Vortisch, P., 2001, 'Use of PTV-Software (VISUM-online) in the Berlin Traffic Management Center', 11th PTV Vision UGM

Webster, Kuwahara & Chung, 2005, 'Tactical driver lane changing heuristic using forward search', Proceedings of the Japan Society of Civil Engineers (JSCE) Infrastructure Planning Conference, Miyazaki, Japan

Westerman, M., 1995, 'Real-Time Traffic Data Collection for Transportation Telematics' PhD Thesis, Delft University of Technology

Wolfram, S., 1983, Statistical mechanics of cellular automata, *Rev. Mod. Phys*. 55 601–644

Yang, H. et al, 1994, The equilibrium-based origin-destination matrix estimation problem, *Transportation Research 28B*, 23-33

Yang, Q. & Koutsopoulos, H. N. (1996), 'A microscopic traffic simulator for evaluation of dynamic traffic management systems', *Transportation Research C*, 4(3), 113-129

Yang, Q. and H. Koutsopoulos, 1997, A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research C*

Yousif, S., Hunt, J., 1995, 'Modelling lane utilization on British dual carriageway roads: effects on lane changing', *Traffic Eng. Control* 36 (12), 680-687

Appendix A: Desired speeds from remote sensing data

To determine the distribution of desired speeds, remote sensing data was analyzed. Since most remote sensing data is gathered for the transmission phase between free flow and congestion the choice of datasets was limited. However, the NGSIM project (http://ngsim.camsys.com/) offers individual vehicle trajectory data also for free flow situations.
Figure 55 shows a plot of individual speeds on the Interstate 80 in Emeryville, USA. The observed stretch includes an on-ramp and so the speeds of the vehicles joining the road from the on-ramp are extracted. That leaves a bandwidth of free flow speeds between 75 and 130 km/h with a speed limit of 105 km/h (65 miles/h).

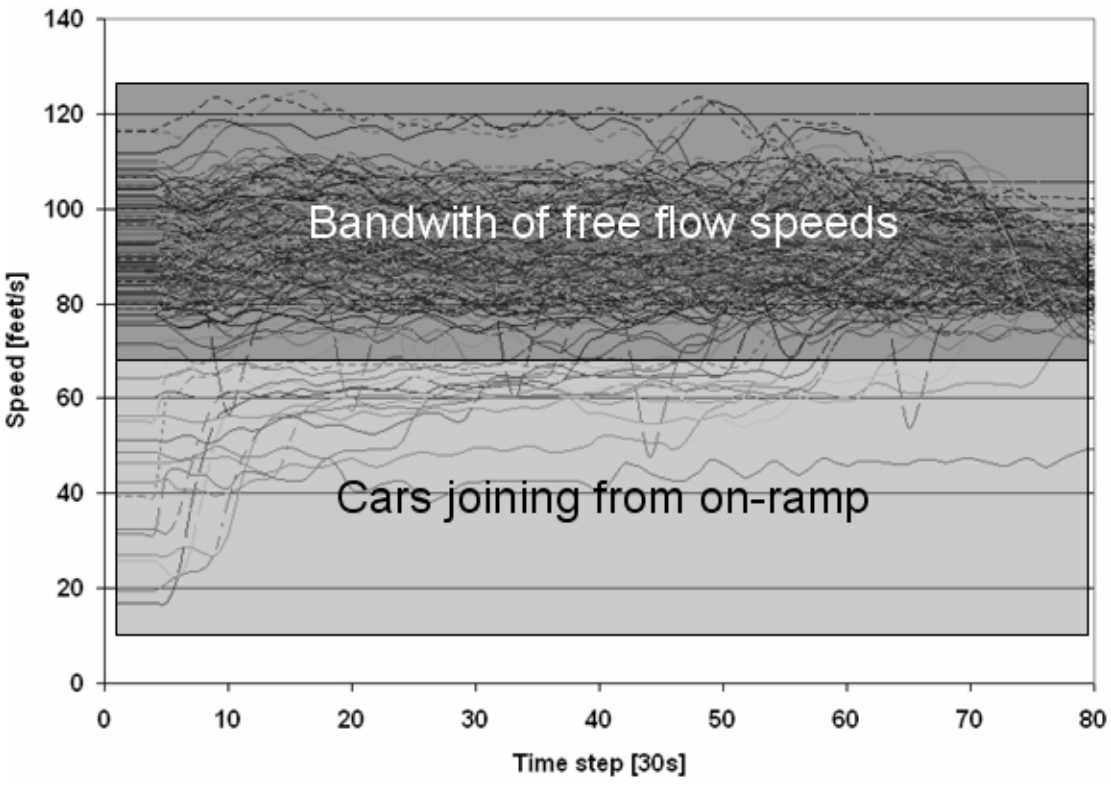

Figure 55: Individual speeds on the Interstate 80 in Emeryville, USA

Based on these individual speed decisions we extracted a distribution of desired speeds for traveling in free flow conditions. The dataset consists of 4733 individual vehicles and the distribution around the speed limit is shown for cars in Figure 56 and for trucks separately in Figure 57.

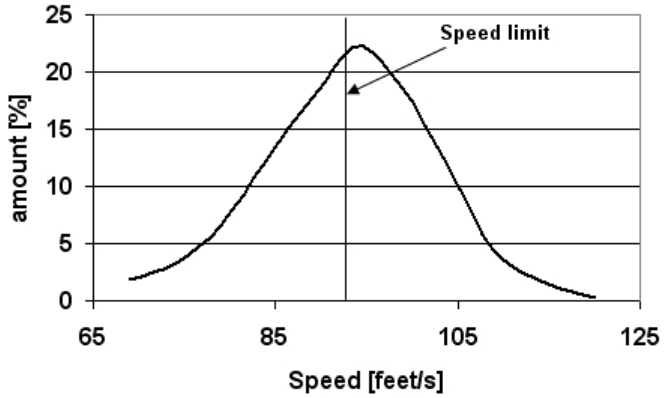

Figure 56: Distribution of free flow speeds from cars

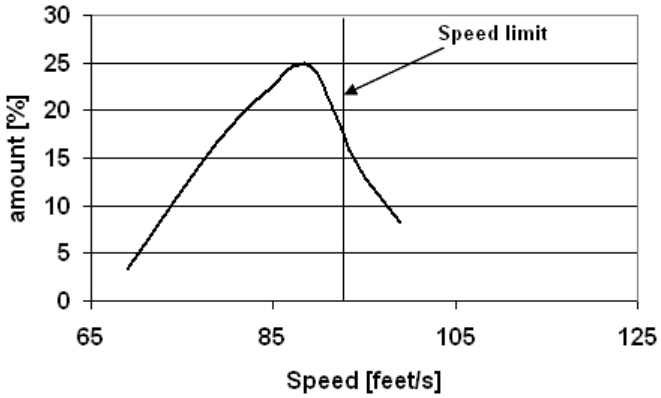

Figure 57: Distribution of free flow speeds from trucks

Hoogendoorn et al. (2004) stated that estimation of free speeds will generally lead to a biased result since the contribution of the high free speed observations are underestimated, due to the fact that a driver with a high free speed is more likely to be constrained than a driver with a low free speed. Therefore, we used their proposed non-parametric Kaplan Meier approach to correct our findings from the remote sensor data.

Appendix B: Transfer Protocols in FrOnT

Since FrOnT is a distributed system it raises the need for communication between the involved software modules. FrOnT uses two types of communication: First the internal communication between the various agents (see 2.4) and second the external communication to the MiOS modules for simulation, traffic control, driving behavior, and OD estimation and prediction. While the first type of communication is handled by the middleware described in 2.6.2 the communication between the MiOS modules is handled with FrOnT protocols. The following gives a short introduction to the concepts of network programming before external data transfer protocols of FrOnT are described.

Network programming in a nutshell

To transfer data from one computer to the other, the data gets encapsulated in a package, which contains a header, the data itself and optional a footer by the protocol used for the transfer (the TFTP protocol). Then the package (TFTP header included) is encapsulated again by the next protocol (UDP), then again by the next (IP), then again by the final protocol on the hardware physical layer (Ethernet). So a simple set of data gets various times encapsulated before transmission. An illustration of this encapsulation is given in Figure 58.

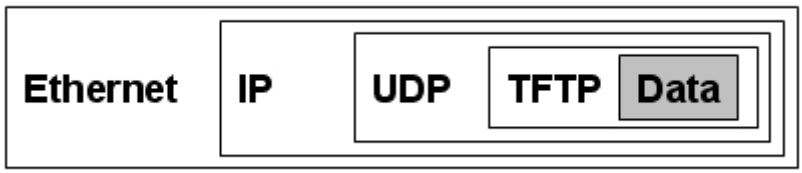

Figure 58: Encapsulation of a dataset before transfer

When another computer receives the packet, the hardware strips the Ethernet header, the kernel strips the IP and UDP headers, the TFTP program strips the TFTP header, and it finally has the data. The full layered network model encapsulates data in the following stages:

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

The Physical Layer is the hardware (serial, Ethernet, etc.). The Application Layer is the most abstract layer and the only important one for the software design of FrOnT, where a socket connection is used, which leaves us carefree about the underlying further network protocol layers.
There are several sockets that can be used. The most common ones are stream socket and datagram sockets, next to several raw sockets, which are very powerful but of less importance.
Stream sockets are reliable two-way connected communication streams. If you output two items into the socket in the order "1, 2", they will arrive in the order "1, 2" at the opposite end. They use a protocol called "Transmission Control Protocol", otherwise known as "TCP". TCP ensures that data arrives sequentially and error-free.
Datagram sockets also use IP for routing, but they don't use TCP; they use the "User Datagram Protocol", or "UDP", they do not maintain an open connection as stream sockets. Data gets build in a packet, encapsulated with an IP header with its destination information, and send out. No connection is needed. They are generally used for packet-by-packet transfers of information.
In FrOnT stream sockets are used and since the underlying transfer of data is handled automatically the following focuses on the different stages of data communication in the FrOnT protocols.

External FrOnT Protocols

The external FrOnT protocols control the data transfer between the framework and the MiOS simulation modules. Stream sockets are used to ensure the availability of all modules and to react immediately on any kind of connection problems during runtime.

Traffic control
The simulator in MiOS itself is equipped with a simple traffic control component, but to enable more sophisticated controller software to incorporate with the simulation a

traffic control server is started with each simulation to exchange information. Here we describe the capabilities of the server side and resolve the specification for clients which can be used with the framework.
The dataflow for the traffic control goes server side along the following lines:

- Waiting for client request
- Sending initialization data
- Wait for confirmation and settings
- Sending detector values
- Receiving traffic light status

When MiOS is started, FrOnT will in parallel startup the traffic control server on port 4444, which will stay in a waiting loop, until a client tries to connect through a stream socket connection. The server is multithreaded and therefore able to handle several clients at the same time. As soon as a client connects, the server is sending an initialization data package, including a list of the defined intersections of the network and the according streams due to the international standard enumeration (see Figure 59).

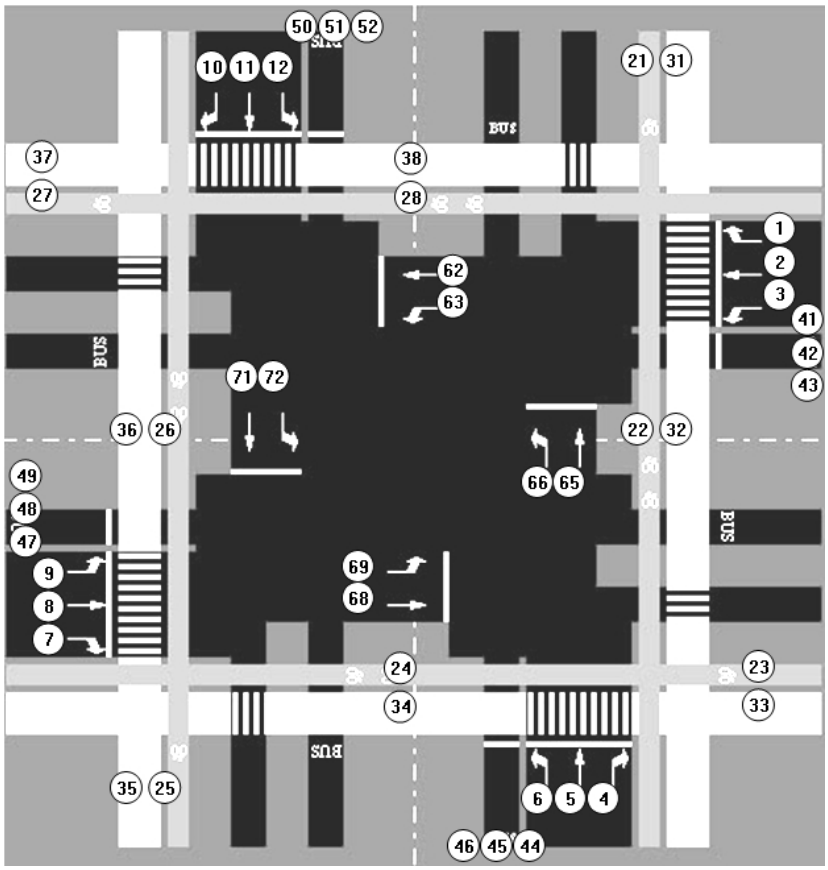

Figure 59: Enumeration of approaching links to an intersection

This initialization looks then as follows:

```
<number of intersections>,
<intersection ID>,
<stream>,<stream>, … <stream>
…
<intersection ID>,
<stream>,<stream>, … <stream>
<end>
```

Then the server waits for the client to reply and to transmit the required time step for updating the network information as a multiple of 1/10 of a second. So if the client is updating every second the response would be:

```
10
<end>
```

After the server gets this response, the status of the detectors for the involved streams gets transmitted as a simple binary value, where a “1” states a vehicle present at the detector and a “0” otherwise. It should be noted that if a vehicle passes a detector and the value is set to “1”, it remains one until the traffic light status changed to green. So a possible transmission could look like:

```
000111010001100000001000000000111
```

The client reacts on the transmission by sending back the new status of the traffic light of the intersection. Since a traffic-light can have more states than on and off, the following values are used: “0” for a green light, “1” for amber, “2” for red, and “3” for blinking amber (traffic light out of service). Note that the length of both transmissions is always the same size. This leads to a response like:

```
102313010201120302003300020000232
```

Therewith, the requirements for a client can be concluded and the order of information interchanges with it:

- Connect to server on port 4444
- Wait for initialization data
- Send confirmation by transmitting the update time step

- Receive detector vales
- Send traffic light status

Due to the fact most standalone traffic controllers will not provide these transmission capabilities, MiOS provides a basic client, written in Java2 and therewith portable to any platform, which establishes the connection. The only work left to do is the data exchange between this client and the control software.

OD estimation

In terms of OD estimation it is not so easy to reduce the actual transmission to a single data string that we used for the traffic control communication. To be consistent, the protocol has a similar structure:

- Waiting for client request
- Sending initialization data
- Wait for confirmation
- Sending measurements (actual), assignment, historical OD, departing cars per time step (simulation), OD separated measurements (simulation)
- Receive new OD estimate and start over time

The OD estimation server is started on port 4445 when MiOS is started. MiOS will use its own mechanism for OD estimation until a client connects to this port and the connection is established. If a client connects, the server will return an initialization dataset with the number of OD pairs and measurement points. The confirmation by the client is completed by sending the time step for the updates. In the following cycle the server will send its information in the following form:

```
<measurement_1>,<measurement_2>, …, <measurement_N>
<histOD_1>, <histOD_2>, …, <histOD_M>
<departsOD_1>, <departsOD_2>, …, <departsOD_M>
Measurement point 1
<arrivalsOD_1>, <arrivalsOD_2>, …, <arrivalsOD_M>
Measurement point 2
<arrivalsOD_1>, <arrivalsOD_2>, …, <arrivalsOD_M>
…
Measurement point N
<arrivalsOD_1>, <arrivalsOD_2>, …, <arrivalsOD_M>
<end>
```

Then the server waits for the response including the new estimated OD table and the number of time steps the simulation should rewind in this format:

```
<OD_1>, <OD_2>, …, <OD_N>
<number of time steps to rewind>
```

To be able to rewind the simulation and to start over with a new OD matrix the server requests to MiOS to create key frames, which are snapshots of the network state at each time step of updating the information for OD estimation.

Driving behavior

The server for the driving behavior is started on port 4446. Due to the amount of vehicles running in a network and the overhead produced by the communication it is recommended to cluster the communication for all vehicles on a link or even an area. Small networks could even be handled with one update request. Since the computational effort for microscopic simulation scales with the individual vehicles to be handled, performance test can help to determine the optimal clustering.

To keep the possibilities for computing the individual driving behaviors the transmission includes the vehicle dynamics as well as a snapshot of the surroundings of the car. Since vehicle dynamics don't change over time, they are just transmitted once in the beginning. The snapshot of the surroundings includes the vehicles speed, acceleration, the occupation of the neighbored lanes (20m, or one car to the back, and 100 meters, or 2 cars to the front), and the traffic rules for the actual link on which the vehicle is traveling. The protocol looks as follows:

- Waiting for client request including number of vehicles
- Sending vehicle dynamics
- Receive confirmation
- Send vehicle surroundings
- Receive new vehicle states

In contrast to the other protocols the client has to send the maximum numbers of vehicles that can be handled before the server takes action. Then the vehicle dynamics are sent according to the number of vehicle classes defined in MiOS:

```
<number of vehicle classes>
<vehicle class 1>
     <max speed>
     <max acceleration>
     <desired acceleration>
     <max deceleration>
```

```
        <desired deceleration>
    <vehicle class N>
        <max speed>
        <max acceleration>
        <desired acceleration>
        <max deceleration>
        <desired deceleration>
```

The confirmation of the client is a respond including the number of vehicle classes received, before the cycle of updates with the following information starts:

```
    <number of vehicles>
    <vehicle_1>
        <vehicle class>
        <actual lane>
        <actual speed>
        <actual acceleration>
        <space in front>
        <average speed in front>
        <space on left lane front>
        <average speed in left lane front>
        <space on right lane front>
        <average speed right lane in front>
        <space on left lane back>
        <speed in left lane back>
        <space on right lane back>
        <speed right lane back>
        <link density>
        <average speed on link>
    <end>
    <vehicle_2>
        …
    <end>
    <vehicle_N>
            …
    <end>
```

After processing the data and computing the driver's actions based on the encountered situation the client returns the new state of the vehicles:

```
    <vehicle_1>
        <new lane>
```

```
        <new speed>
        <new acceleration>
    <vehicle_2>
        …
    <vehicle_N
        …
    <end>
```

With this information the network state in the simulator can be updated. It should be noted that in case of a connection failure the simulation is halted, since there is no fallback procedure as in case of OD estimation and traffic control. However, if a connection fails, FrOnT will use if possible another available client in the framework. That implies that by running more clients than needed for each module increases the robustness and stability of the system.

Appendix C: Data description files in FrOnT

The purpose of using data description files is to be able to include all different kind of datasets without explicitly hard coding them into the software. The concept of data description files in FrOnT is based on the interface definition language (IDL), specified by the Object Management Group (OMG).
The OMG (www.omg.org) is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Their membership includes virtually every large company in the computer industry, and hundreds of smaller ones. Most of the companies that shape enterprise and Internet computing today are represented on the Board of Directors. The flagship specification is the multi-platform Model Driven Architecture (MDA), recently underway but already well known in the industry. It is based on the modeling specifications the meta-object facility (MOF), the unified modeling language (UML), XML metadata interchange (XMI), and the common warehouse meta-model (CWM). OMG's own middleware platform is CORBA. All specifications are available for download at the OMG website.
In the following the technique of the IDL is described and ported from software coupling to data handling before giving an example of the usage in FrOnT.

OMG Interface definition language for distributed systems

The OMG describes the concept of IDL as follows: *"Encapsulation is the principle of object-orientation that enables seamless distribution. By encapsulating the internal structure and mechanism of objects inside a boundary that the client is not allowed to penetrate, not only flexibility but also simplicity of architecture that could not be achieved any other way is gained. OMG IDL is used to define the interface that sits on the outside of the boundary, allowing the only communication that the object conducts with the outside world.*
For distribution to work in this architecture, we need to define interfaces that both client and server object understand and can use easily, regardless of their platform, operating system, programming language, network connection, or other characteristics. An interface definition must specify the operation to be performed and all of the input and output parameters with their types, allowing client and server to encode and decode values for their travel over the network. And finally, since it's

always possible for something to go wrong when we make an invocation, the language should support robust exception handling" (OMG website).

In practice the IDL works at two levels: First, every object has a type name, which is the same as the interface name you assign in its IDL declaration, the operations that it can perform and the variables that it understands. Second, IDL variables are typed based on the official list of types allowed by the IDL specification. Either fixed or variable length can be defined. The IDL gets compiled into client stubs and object skeletons. Stubs and skeletons serve as proxies for clients and servers, respectively (see Figure 60).

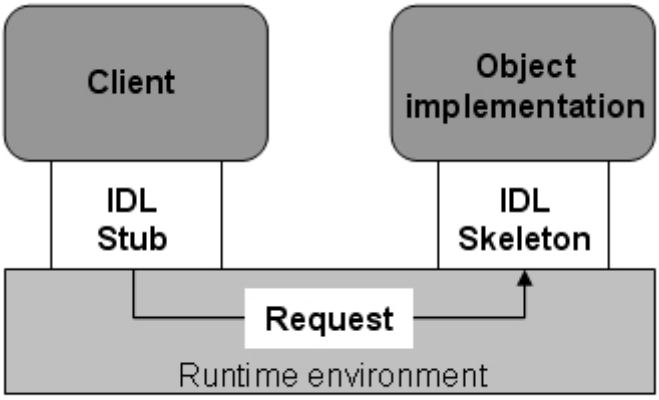

Figure 60: Working scheme of remote method invocation using IDL

Clients access objects only through their interface, invoking only those operations that that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation. Because IDL defines interfaces so strictly, the stub on the client side has no trouble meshing perfectly with the skeleton on the server side, even if the two are compiled into different programming languages, or even running on different environments from different vendors.

Porting IDL to a data description file

Now we are going to port the concept from software coupling to data handling. The idea is to use a flexible object that can be filled with various kinds of measurement data, but can be handled in FrOnT consistently without changes in the software itself. This requires the implementation of a *data object,* described by a data description file

(DDF), which feeds a FrOnT measurement object (see 2.3.3) and so can be used by clients throughout the system. The concept is shown in Figure 61.

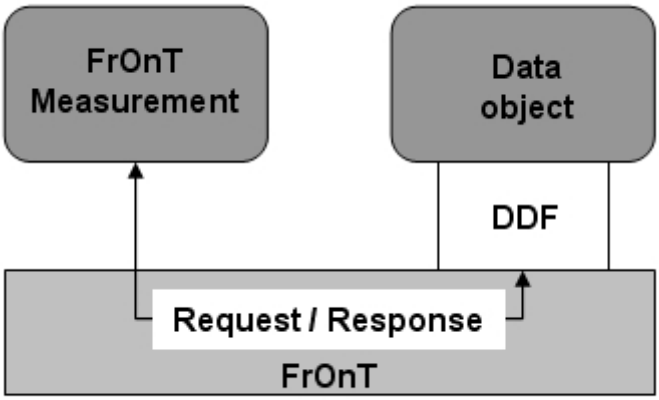

Figure 61: Working scheme of data gathering using DDF

The FrOnT measurement object refreshes its contents by requesting new data values from the flexible data object via the FrOnT runtime environment. The request is parsed with the help of the DDF and the response is generated in the FrOnT measurement format.
To be able to parse the request the DDF has to describe the variable, methods and handling of the data object. For further simplification, the data object is fitted with generic methods that get filled with commands created from the DDF on the fly.

The data object with its variable and methods

The data object consists of five variables and six methods. The variables are the following:

- ID (to map the data object to the FrOnT measurement object)
- Data_location (internal, external)
- Data_type (text, database)
- DDF (reference to the data description file)
- Values (Vector to be filled with the measurements)

While the first four variables are fixed values, the vector for the values can be filled with objects of any kind and size. So when creating an instance of the data object, no information of the amount or kind of data needs to be known. The methods of the data object are next to the constructor, which sets the first four variables:

- Check_data_connection(), which checks depending on the data location and type if the database or the text file is reachable without checking any contents
- Open_data_connection(), which establishes the connection of the object with the data source
- Close_data_connection() to terminate the connection to the data source
- Receive_measurements() to receive new measurement information from the data source
- Transform_data() to transform the data from the values vector to the FrOnT measurement format
- Get_data(time, location), the method called by the FrOnT measurement object to update its own values for a given time step.

With these methods being strongly dependent on the source and the representation of the measurement data at the source the DDF is used during runtime to create the needed SQL queries to the database or to read the text files properly. Therefore, the DDF contains the following information:

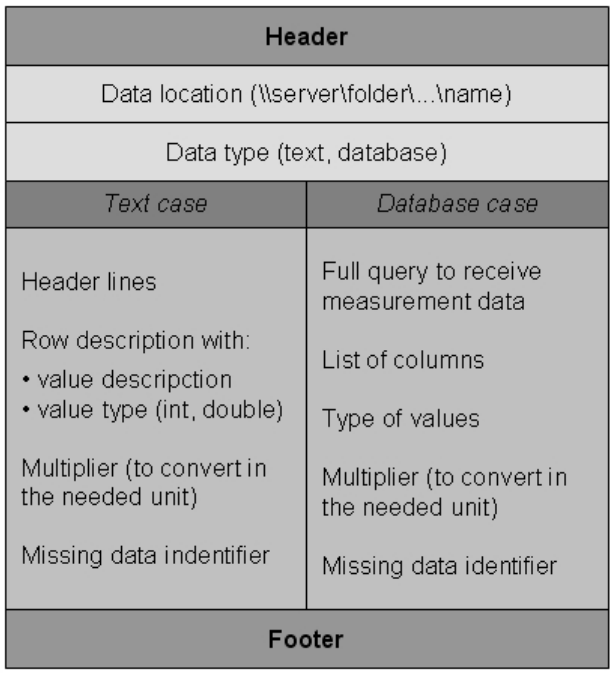

Figure 62: Structure of a simplified data description file (DDF)

Header and footer of the data description file are used for consistency and version check as well as for mapping the database columns to the value description understood in FrOnT.
To clarify the process an example is given in the following for retrieving measurement data from the Regiolab Delft server.

Example of a DDF for the Regiolab Delft data

The Regiolab Delft server is a database server with a web-interface. The data location is therefore a web address:

http://www.regiolab-delft.nl/monica/

with the data type being a database. Now the database specific information is given with the information about the retrieving of data and its format. In case of the Regiolab Delft server this query would look like the following:

*retrieve.cgi?fromdate=<**startdate**>&todate=<**enddate**>&fromtime=<**starttime**>&totime=<**endtime**>&aggregate=1&roadnumber=<**locationA**>&direction=R&mindist=<**locationB**>&maxdist=<**locationC**>&dvkletter=&mflag=&format=&retrieve=Submit+query*

The bold values in this query are inserted during runtime, while the rest of the query is fixed. After processing the query is:

*retrieve.cgi?fromdate=**20060101**&todate=**20060101**&fromtime=**0700**&totime=**0701**&aggregate=1&roadnumber=**13**&direction=R&mindist=**5500**&maxdist=**5600**&dvkletter=&mflag=&format=&retrieve=Submit+query*

Additional the header and footer have to be specified. The header starts with the version number and is followed by a mapping table for the retrieved data:

```
*******************************************
FrOnT version 0.9.8.12
*******************************************
Mapping table:
Column1 -> Detector_ID
Column2 -> Detector_Location
Column3 -> Link_Number_of_Lanes
```

```
Column4 -> Measurement_Date
Column5 -> Measurement_Time
Column6 -> Measurement_Reliable
Column7 -> Measurement_Flow
Column8 -> Measurement_Speed
*****************************************
```

The header is then followed by the body information:

```
http://www.regiolab-delft.nl/monica/
Database
*****************************************
retrieve.cgi?fromdate=<startdate>&todate=<enddate>&fro
mtime=<starttime>&totime=<endtime>&aggregate=1&roadnum
ber=<locationA>&direction=R&mindist=<locationB>&maxdis
t=<locationC>&dvkletter=&mflag=&format=&retrieve=Submi
t+query
*****************************************
Column1, Column2, Column3, Column4, Column5
Column6, Column7, Column8
*****************************************
String, String, String, Date, Time, char, int, int
*****************************************
1
*****************************************

*****************************************
```

And the DDF is finalized with the footer, giving a checksum of the included lines and again the version number.

```
Checksum: #31
FrOnT version 0.9.8.12
*****************************************
```

With this information completed, the data object is able to retrieve the information and to convert it to the FrOnT measurement format, so that it can be used throughout the system.

Summary

Traffic management nowadays is a complex task where pure monitoring of the network is not sufficient anymore for network operation. To support the road authorities and traffic engineers to optimize the traffic operation models are developed to estimate traffic states and to predict future traffic states in advance to give the possibility to anticipate with the traffic and not just to react on it. Various models from traffic modeling, control, and mathematics are building a suite of tools to support traffic engineers in the optimization process. These tools are developed by experts in different areas and this thesis deals with the design and development of a framework that enables the user to combine these tools to a system with a single user interface and the flexibility of changing modules independent from the others. This allows researchers to test new approaches and algorithms in a stable environment and allows practitioners to use state of the art modules as soon as they are available without a time consuming reengineering of the existing system or changes in the system hardware.

The basic input for online traffic management systems are real-time measurements, consisting of traffic conditions, traffic control messages, controller programs and weather information. These measurements come from different systems on the roadside and are stored in specific formats and locations. To cope with this variety, we developed a database structure to store the collected data in a common structure so that the handling of the data can always be performed in the same way. At a further step the database is extended so that not only the measurements, but also the network itself can be stored with all elements. Therewith, we created a single data source for traffic management systems.

Storing the data is just one process, but before that, the data has to be gathered from various locations. Since the measurements are collected at the roadside they get transmitted to a storage location. This indicates a distribution in a computer network. From Internet applications the usage of bots (software tools that search the network for information autonomously) is well known to discover information in the World Wide Web. This principle we used in our Framework for Online Traffic management (FrOnT) to gather data from different locations throughout a network. Collector Agents are gathering continuously the data from defined locations and convert them into the internal FrOnT format for further usage. The network definition on the other hand is user controlled and done by a network editing tool designed for FrOnT.

The data gathered and stored now needs to be processed to be used as supporting information for traffic management operations. Since the aim is to distribute the process and to change every component of the system separately we introduce another

agent to control and synchronize different processing modules. These Control Agents use the stored data to feed attached simulation tools with the input they need. The communication between the Control Agent and the processing modules is done by socket connections which are independent from the used hardware or programming language and use the TCP/IP protocol as a common way of transmission. This concept allows the incorporation of autonomous modules to a system, but limits the performance of the system to the computational power of a single machine. Considering large scale networks and control systems for big cities this is not acceptable. Therefore, we decided to distribute the computation load throughout a computer network. This allows parallel processing without the need of the using parallel algorithms. The extension of FrOnT is based on a mainframe with a Control Agent and a lookup table for modules and other FrOnT runtime environments in the network. To establish the communication between the platforms, we introduce Communication Agents to pass the information without adding computation load to a Control Agent. In this way, FrOnT can be distributed to various machines and the processing can be accelerated. Distributed systems rely on the accessibility of all involved computers and are vulnerable to hardware or communication errors. To cope with this problem and to make FrOnT as robust as possible, a Recovery Agent is mirroring the complete communication and buffers it. In the incident of a malfunction of one involved system, the mainframe can look up and address a different machine with the task and with help of the Recovery Agent reset the whole system to the last conflict-free state. If the main platform encounters a problem, another node becomes main platform. This requires a startup of the lookup services and a system wide update of communication lines, which takes time, but let the system "repair" itself up to the point where all nodes break down. Therefore, the FrOnT system is very robust and increases the robustness by distribution in a network.
So far the Front system, but to use it for traffic management it needs the modules for:

- simulation,
- driving behavior,
- OD estimation, and
- route choice.

This led to the development of MiOS (Microscopic Online Simulator). MiOS is a suite of autonomous working modules to perform a microscopic online simulation. For the simulation we have chosen a cellular automata (CA) model due to the fast programming. Extension to the CA model used in MiOS is the reduction of the cell length to 0.5 m to allow a better representation of distances and gaps and to allow the simulation of heterogeneous traffic consisting of cars, vans, trucks and so on. This means that vehicles are not occupying a single cell of the automaton but span over

several cells according to the vehicle length. With the minimum movement of a vehicle of one cell per update, the time step has been set to 0.1 seconds which corresponds to a speed discretization of steps in 18 km/h. Usually CA models consist of transmission rules to simulate the vehicles. However, this is in contrast to the idea of individual and autonomous modules. So, the update rules are not integrated any longer in the simulator, but in a separate driving behavior module.

By extracting the driving behavior there was no need to follow the concept of transmission rules. Triggered by the literature on driving behavior models and the conclusion drawn from them, we decided to implement a new approach of behavior modeling. Instead of setting up a mathematical or rule based model and to calibrate it with measurement data, we started with the measurement data from microscopic observations to train a Bayesian network. This technique provided us with distributions of individual driving behavior in experienced surroundings. The perception of the driver used as input for the simulator allows a pattern matching procedure to determine the reaction of a driver in the actual situation. To include the fact that humans are not able to perceive their surroundings in terms of exact distances, and speed differences the accurate perception from the simulator is combined with a noise function based on the observation and the reaction based on this erroneous perception is the feedback to the simulator.

The third module is matching local traffic counts to an OD pattern used by the vehicle generator of the simulator. The field of OD estimation is very complex and for practical reasons, MiOS is using a basic OD estimation and prediction based on predefined OD matrices for different periods of the day (night, morning peak, noon, and evening peak). These matrices are multiplied by vectors determined from the measurements to adapt to the actual traffic demand. This method is very basic and requires knowledge about the existing OD relation in the network. Even though we did not give more attention to the OD estimation process in this thesis we developed the interface according to the needs of state of the art OD estimators.

Based on the estimated OD matrix at the simulation time the simulator generates vehicles to enter the network. The route choice of the vehicles from their origin to their destination is described by a route choice model. The route choice model of MiOS is using a link cost function based on the actual travel time, route guidance information and road type to determine the path of the vehicle. After an evaluation of the whole network, the travel costs are calculated and a shortest path search is performed. Vehicles do not necessarily stick to their initial route choice. We extended the model with a comfort factor for each driver. This factor describes the willingness of a driver to switch routes on the journey. This factor is based on the link density and the difference between the desired speed and the actual speed of the driver. If this comfort factor reaches a driver specific threshold the vehicle will be redirected to a new route to its destination.

With these modules connected to FrOnT it is possible to run a microscopic online simulation of a given network. As a test bed we used the region of the City of Delft. This network consists of major inner city roads, provincial arterials through the city and two motorways surrounding the city. The real-time measurements are taken from the Regiolab Delft, which is a combined research effort of the Delft University of Technology, the research school TRAIL, the Dutch Ministry of Transport, the province of South Holland, the municipality of Delft and the traffic industry. Regiolab Delft provides detections from double loop detectors on the motorways, and single loop detections and camera observation in the city itself. We tested the capabilities of the FrOnT – MiOS system separately on the motorway network and the urban network by predicting travel times with a horizon of 30 minutes ahead. In a second test we run the system on the combined network and simulated a day on which a major incident led to a full blockage of one of the motorway. The system was able to predict the reaction of the traffic and the new distribution of vehicles through the network.

We have proven that FrOnT in connection with the autonomous modules for simulation of MiOS is able to support traffic engineers in their task of network operations and designed a system that allows to serve as a test environment for newly developed algorithms under real life conditions. The possibility of using state of the art algorithms without reengineering the entire system gives flexibility to researchers and practitioners that has the potential to decrease the time span between development and implementation of methods significantly and opens the way for a standardization of communication protocols among traffic engineering applications.

Samenvatting

Verkeersmanagement is tegenwoordig een complexe taak waar meer voor nodig is dan het slechts waarnemen van verkeer in een verkeersnetwerk. Ter ondersteuning van de wegbeheerders en verkeersspecialisten worden verkeersmodellen ontwikkeld die de toestand van een verkeersnetwerk adequaat kunnen schatten en voorspellen. Dit biedt de mogelijkheid om niet alleen tijdig reageren op verkeerscondities maar ook te anticiperen op toekomstige situaties en op de reactie van het verkeer op maatregelen. Verkeerskundige applicaties, ontwikkeld vanuit verschillende achtergronden ondersteunen de verkeerskundige in het optimalisatieproces van verkeersmaatregelen. Deze verkeerskundige applicaties worden ontwikkeld door experts uit verschillende gebieden.
Dit proefschrift behandelt het ontwerp en de ontwikkeling van een raamwerk dat de gebruiker in staat stelt deze applicaties te combineren in één enkele gebruikersinterface, met de flexibiliteit om verschillende modules te wisselen. Een dergelijke aanpak stelt onderzoekers in staat om nieuwe methodes en algoritmes te testen in een stabiele omgeving waarin *state-of-the-art* modules geïntegreerd kunnen worden zonder een tijdsintensief proces van herontwerp of veranderingen in de hardware van het systeem.

De basis invoer voor *online* verkeersmanagement wordt geleverd door *real-time* metingen bestaande uit de verkeerscondities, weerinformatie, beheersmaatregelen en berichten. Deze metingen zijn afkomstig van verschillende wegkantsystemen waar zij in een eigen formaat zijn opgeslagen. Om met deze heterogeniteit om te gaan is een databasestructuur ontwikkeld waarin deze data op een algemene wijze kan worden vastgelegd. Zo kunnen de bewerkingen op deze data op een structurele en universele wijze plaatsvinden. In een volgende stap wordt de database uitgebreid zodat niet alleen de metingen, maar ook het netwerk op universele wijze kan worden opgeslagen. Zodoende wordt er één enkele databron ontwikkeld voor verschillende verkeersmanagementsystemen.
Voordat de data kan worden opgeslagen moet deze eerst worden opgehaald bij de verschillende bronnen. Deze data wordt vanaf het waarnemingssysteem bij de weg verstuurd naar een opslaglocatie. Dit geeft al aan dat het gaat om een gedistribueerd computernetwerk. Binnen verschillende internetapplicaties is het gebruik van een "agent" (een software programma dat het netwerk zelfstandig afzoekt naar informatie) zeer gangbaar om informatie op het "World Wide Web" te zoeken. Dit principe is ook gebruikt in het raamwerk FrOnT (*Framework for Online Traffic Management*) om data van verschillende databronnen te verzamelen. Hierin zijn "verzamel" *agents* continu bezig met het verzamelen van gegevens en deze te converteren naar het

interne FrOnT-formaat voor verder gebruik. De definitie van het verkeersnetwerk daarentegen wordt door de gebruiker zelf ingevoerd met behulp van een intern bewerkingsprogramma in FrOnT.
De verzamelde en opgeslagen gegevens, dienen vervolgens te worden verwerkt om het als ondersteunende informatie te gebruiken voor verkeersmanagement-toepassingen. Omdat het doel is dit proces te distribueren en iedere verwerkingsmodule binnen het systeem vervangbaar te maken, wordt er nog een extra *agent* geïntroduceerd ter besturing en synchronisatie van deze verschillende verwerkingsmodules. Deze *besturings-* of *control -agents* gebruiken de opgeslagen data in FrOnT om de verkeerssimulator te voorzien van de benodigde data.
De communicatie tussen de *control-agent* en de verschillende verwerkingsmodules vindt plaats door middel van *socket connections* die onafhankelijk zijn van de gebruikte hardware of programmeertaal. Deze *sockets* gebruiken het TCP/IP protocol voor communicatie.
Een dergelijke aanpak maakt het mogelijk om zelfstandige modules in het systeem te integreren, maar beperkt deze tegelijkertijd tot de rekenkundige belasting van één enkele computer. Bij grootschalige verkeersnetwerken of stedelijke netwerken is dit echter een onacceptabele beperking. Daarom is er besloten om de rekenkundige belasting over een computernetwerk te distribueren, wat parallel rekenen mogelijk maakt, zonder de noodzaak tot het gebruik van parallelle algoritmes.

Deze uitbreiding van FrOnT is gebaseerd op een *mainframe* met een *control-agent* en een *lookup* tabel voor modules en andere FrOnT *runtime environments* in het netwerk. Om communicatie tussen de verschillende platformen mogelijk te maken zijn er *communicatie-agents* geïntroduceerd die deze informatie communiceren zonder dat dit ten koste gaat van additionele rekenkundige belasting voor de *control-agents*. Op deze wijze kan FrOnT gedistribueerd worden over verschillende machines en kan het rekenen worden versneld. Gedistribueerde systemen zijn afhankelijk van de toegankelijkheid van alle betrokken computers en zijn kwetsbaar ten aanzien van hardware of communicatie storingen. Om ook dit probleem op adequate wijze op te lossen, en daarmee FrOnT zo robuust mogelijk te maken, is er een *herstel-agent* geïntroduceerd. Deze *agent* kopieert alle communicatie en slaat deze op. In het geval van een storing van een van de betrokken computers, kan het *mainframe* een andere computer in het netwerk opzoeken en deze met de taak van het verstoorde systeem belasten. Met behulp van de *herstel-agents* wordt het gehele systeem teruggezet naar de laatst bekende storingsvrije toestand. Indien het hoofdplatform met een probleem geconfronteerd wordt, zal een andere computer in het netwerk deze taak op zich nemen. Bij een dergelijk falen moeten de *lookup services* opnieuw worden gestart en alle communicatie lijnen in het hele systeem vernieuwt. Een dergelijke operatie kost tijd, maar stelt het systeem in staat om zich te herstellen tot het punt voorafgaand aan het falen. Op deze wijze is FrOnT een robuust systeem, en verhoogt deze de robuustheid verder door distributie in een netwerk.

Om het FrOnT systeem te gebruiken voor verkeersmanagement zijn de volgende modules nodig:

- Simulatie;
- Rijgedrag modelleren;
- herkomsten en bestemmingen (HB) schatten;
- routekeuze.

Deze benodigdheden hebben geleid tot de ontwikkeling van MiOS (*Microscopic Online Simulator*). MiOS is een pakket van zelfstandig werkende modules met als doel een *online* verkeerssimulatie mogelijk te maken. De simulatie is gebaseerd op een *cellular automata* (CA) aanpak. Daarvoor is gekozen vanwege de aantrekkelijke hoge rekenkundige snelheid die deze aanpak biedt. Een van de uitbreidingen aan het CA-model in MiOS is de reductie van de cellengte tot 0.5 meter, wat een betere representatie van (volg)afstanden en heterogeen verkeer (auto's, vrachtwagens en bestelvoertuigen) mogelijk maakt. Voertuigen beslaan niet één enkele cel van de *automata* maar verschillende cellen, al naar gelang de voertuiglengte. Omdat de minimum beweging van een voertuig één cel per update is, is de tijdstap van de simulatie op 0.1 seconde gezet. Dit maakt het mogelijk snelheidsverschillen tot 18 km/uur te onderscheiden. Normaliter gebruiken CA modellen transmissieregels om de beweging van voertuigen te simuleren. Dit past echter niet bij het concept van individuele, verwisselbare, autonome modules. Daarom is gekozen om de transmissieregels niet in de simulator te integreren, maar in een aparte module die verantwoordelijk is voor de simulatie van het rijgedrag.

Op basis van de bestaande literatuur over rijgedragmodellen en de conclusies die hieruit getrokken werden, is besloten tot een nieuwe aanpak ten aanzien van rijgedragmodellering. In tegenstelling tot het formuleren van een rekenkundig of regelgebaseerde model en dit vervolgens te kalibreren met meetgegevens, zijn microscopische meetgegevens als basis genomen om een Bayesian netwerk mee te trainen. Deze techniek leverde vervolgens verdelingen op van individueel rijgedrag in de door gebruikers ervaren rijomgeving. Met perceptie van de automobilist als invoer in de simulator is het mogelijk een patroonherkenningprocedure uit te voeren, waarmee de reactie van de automobilist in de huidige situatie kan worden bepaald. Om te compenseren voor het feit dat automobilisten niet in staat zijn hun exacte volgafstand en snelheidsverschil in te schatten, worden deze gegevens uit de simulator gecombineerd met een random verstoring gebaseerd op de waargenomen data. De reactie gebaseerd op deze met ruisverstoorde perceptie vormt de gebruikte feedback in de rijsimulator.

De derde module is erop gericht om locale metingen in overeenstemming te brengen met het Herkomst-Bestemmingspatroon (HB-patroon) dat wordt gebruikt in de simulator om de omvang van de voertuigstromen te simuleren. Het schatten van HB-matrices vormt een uitgebreid en complex onderzoeksgebied waarop nog veel ontwikkeling plaatsvindt. Vanuit praktisch oogpunt is ervoor gekozen om een eenvoudige HB-schatter en -voorspeller te gebuiken, gebaseerd op vooraf bepaalde (*a-priori*) HB-matrices voor verschillende periodes (nacht, ochtend spits, middag en de avond spits). Deze *a-priori* matrices worden vervolgens vermenigvuldigd met vectoren die gebaseerd zijn op meetgegevens om daarmee de HB-matrices in overeenstemming te brengen met de huidige verkeersvraag. Deze eenvoudige methode vereist de beschikbaarheid van *a-priori* HB-relaties in het netwerk. Ondanks dat er niet meer aandacht is gegeven aan het HB-schattingsproces in dit proefschrift, is de interface van deze module ontwikkeld om te voldoen aan de eisen van de huidige *state-of-the-art* HB-schatters. Gebaseerd op de HB-matrix behorende bij de simulatietijd, zal de simulator voertuigen genereren die vervolgens het netwerk inkomen. De keuze van de route tussen de oorsprong en de bestemming wordt beschreven door een routekeuzemodel. Het in MiOS geïmplementeerde routekeuzemodel gebruikt kostenfuncties per wegvak, gebaseerd op de huidige reistijden, routegeleidingsinformatie en wegtype om de route per voertuig te bepalen. Na doorrekening van een netwerk, kunnen de kosten worden berekend aan de hand van een kortste-pad algoritme. Voertuigen blijven niet noodzakelijkerwijs gebruik maken van de oorspronkelijk gekozen route. Het model is namelijk uitgebreid met een comfortfactor voor elke automobilist. Deze factor beschrijft de bereidwilligheid van een automobilist om van route te verwisselen gedurende de reis. Deze factor is gebaseerd op de wegvakdichtheid en de verschillen tussen de gewenste en de werkelijke snelheid van de automobilist. Indien de comfortfactor een gebruikersspecifieke grenswaarde bereikt, zal het voertuig een nieuwe route naar zijn bestemming krijgen toegewezen.

Met bovenstaande beschreven modules operationeel binnen FrOnT is het mogelijk een microscopische, online simulatie van een verkeersnetwerk te maken. Als testomgeving is het netwerk van Delft gebruikt. Dit netwerk bestaat uit enkele hoofdwegen door de stad, een provinciale hoofdweg en twee snelwegen die de tegen de stad aan liggen. De metingen op dit netwerk zijn afkomstig van Regiolab Delft, wat een gecombineerd onderzoeksinitiatief is van de TU Delft, onderzoeksschool TRAIL, het Nederlandse ministerie van V&W, de provincie Zuid Holland, de gemeente Delft en de verkeersindustrie. Regiolab Delft geeft toegang tot dubbele-lus gegevens van de snelwegen en enkele-lus en cameradetectie gegevens uit de stad. De mogelijkheden van het FrOnT – MiOS systeem zijn afzonderlijk getest op het binnenstedelijke netwerk en het snelwegen netwerk door op deze netwerken reistijden voor de komende 30 minuten te voorspellen. In een tweede test is het systeem getest aan de hand van detectordata op een dag dat een groot incident plaats had gevonden in

het netwerk, wat leidde tot een volledige blokkade op één van de snelwegen. Het systeem bleek in staat om de reactie van de reizigers en de nieuwe verdeling van voertuigen in het netwerk te reconstrueren.

Het onderzoek toont aan dat FrOnT in combinatie met de autonome modules voor simulatie (MiOS) in staat zal zijn om verkeerskundigen te ondersteunen in hun taak en dat het ontworpen systeem gebruikt kan worden als een testomgeving voor nieuw ontwikkelde algoritmes in werkelijke verkeerscondities. De mogelijkheid die geboden wordt om *state-of-the-art* algoritmes te gebruiken zonder dat het hele systeem moet worden herschreven biedt onderzoekers en mensen uit de praktijk een aantrekkelijke vorm van flexibiliteit. Het geeft de mogelijkheid de benodigde tijd tussen ontwikkeling en implementatie van nieuwe methodes significant te verminderen en het opent de deuren voor een gestandaardiseerde vorm van communicatie protocollen tussen verkeersgerelateerde applicaties.

About the Author

Marc Philipp Miska was born on the 1st of July 1975 in Bitburg, Germany. After finishing a master in Applied Computer Science in Civil Engineering at the University of Hannover, he started a research fellowship at the Franzius Institute for Hydraulics, Waterways and Coastal Engineering at the University of Hanover. His major research interests were telematic systems for inland vessels, traffic management on inland waterways, hinterland transport in North Germany, and container transshipment.
In 2002 he joined as a research engineer the Transportation and Planning Section of the Delft University of Technology to perform a PhD research on Real Time Microscopic Simulation.
Today he works as a research fellow at the Kuwhara Laboratory for Traffic Engineering, part of the Institute of Industrial Science of the University of Tokyo.

TRAIL Thesis Series

A series of The Netherlands TRAIL Research School for theses on transport, infrastructure and logistics.

Nat, C.G.J.M., van der, *A Knowledge-based Concept Exploration Model for Submarine Design*, T99/1, March 1999, TRAIL Thesis Series, Delft University Press, The Netherlands

Westrenen, F.C., van, *The Maritime Pilot at Work*: E*valuation and Use of a Time-to-boundary Model of Mental Workload in Human-machine Systems*, T99/2, May 1999, TRAIL Thesis Series, Eburon, The Netherlands

Veenstra, A.W., *Quantitative Analysis of Shipping Markets,* T99/3, April 1999, TRAIL Thesis Series, Delft University Press, The Netherlands

Minderhoud, M.M., *Supported Driving: Impacts on Motorway Traffic Flow*, T99/4, July 1999, TRAIL Thesis Series, Delft University Press, The Netherlands

Hoogendoorn, S.P., *Multiclass Continuum Modelling of Multilane Traffic Flow*, T99/5, September 1999, TRAIL Thesis Series, Delft University Press, The Netherlands

Hoedemaeker, M., *Driving with Intelligent Vehicles: Driving Behaviour with Adaptive Cruise Control and the Acceptance by Individual Drivers*, T99/6, November 1999, TRAIL Thesis Series, Delft University Press, The Netherlands

Marchau, V.A.W.J., *Technology Assessment of Automated Vehicle Guidance - Prospects for Automated Driving Implementation*, T2000/1, January 2000, TRAIL Thesis Series, Delft University Press, The Netherlands

Subiono, *On Classes of Min-max-plus Systems and their Applications*, T2000/2, June 2000, TRAIL Thesis Series, Delft University Press, The Netherlands

Meer, J.R., van, *Operational Control of Internal Transport*, T2000/5, September 2000, TRAIL Thesis Series, Delft University Press, The Netherlands

Bliemer, M.C.J., *Analytical Dynamic Traffic Assignment with Interacting User-Classes: Theoretical Advances and Applications using a Variational Inequality Approach,* T2001/1, January 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Muilerman, G.J., *Time-based logistics: An analysis of the relevance, causes and impacts,* T2001/2, April 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Roodbergen, K.J., *Layout and Routing Methods for Warehouses*, T2001/3, May 2001, TRAIL Thesis Series, The Netherlands

Willems, J.K.C.A.S., *Bundeling van infrastructuur, theoretische en praktische waarde van een ruimtelijk inrichtingsconcept*, T2001/4, June 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Binsbergen, A.J., van, J.G.S.N. Visser, *Innovation Steps towards Efficient Goods Distribution Systems for Urban Areas,* T2001/5, May 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Rosmuller, N., *Safety analysis of Transport Corridors*, T2001/6, June 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Schaafsma, A., *Dynamisch Railverkeersmanagement, besturingsconcept voor railverkeer op basis van het Lagenmodel Verkeer en Vervoer*, T2001/7, October 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Bockstael-Blok, W., *Chains and Networks in Multimodal Passenger Transport. Exploring a design approach*, T2001/8, December 2001, TRAIL Thesis Series, Delft University Press, The Netherlands

Wolters, M.J.J., *The Business of Modularity and the Modularity of Business*, T2002/1, February 2002, TRAIL Thesis Series, The Netherlands

Vis, F.A., *Planning and Control Concepts for Material Handling Systems*, T2002/2, May 2002, TRAIL Thesis Series, The Netherlands

Koppius, O.R., *Information Architecture and Electronic Market Performance,* T2002/3, May 2002, TRAIL Thesis Series, The Netherlands

Veeneman, W.W., *Mind the Gap; Bridging Theories and Practice for the Organisation of Metropolitan Public Transport*, T2002/4, June 2002, TRAIL Thesis Series, Delft University Press, The Netherlands

Nes, R. van, *Design of multimodal transport networks, a hierarchical approach,* T2002/5, September 2002, TRAIL Thesis Series, Delft University Press, The Netherlands

Pol, P.M.J., *A Renaissance of Stations, Railways and Cities, Economic Effects, Development Strategies and Organisational Issues of European High-Speed-Train Stations*, T2002/6, October 2002, TRAIL Thesis Series, Delft University Press, The Netherlands

Runhaar, H., *Freight transport: at any price? Effects of transport costs on book and newspaper supply chains in the Netherlands*, T2002/7, December 2002, TRAIL Thesis Series, Delft University Press, The Netherlands

Spek, S.C., van der, *Connectors. The Way beyond Transferring*, T2003/1, February 2003, TRAIL Thesis Series, Delft University Press, The Netherlands

Lindeijer, D.G., *Controlling Automated Traffic Agents*, T2003/2, February 2003, TRAIL Thesis Series, Eburon, The Netherlands

Riet, O.A.W.T., van de, *Policy Analysis in Multi-Actor Policy Settings. Navigating Between Negotiated Nonsense and Useless Knowledge*, T2003/3, March 2003, TRAIL Thesis Series, Eburon, The Netherlands

Reeven, P.A., van, *Competition in Scheduled Transport*, T2003/4, April 2003, TRAIL Thesis Series, Eburon, The Netherlands
Peeters, L.W.P., *Cyclic Railway Timetable Optimization*, T2003/5, June 2003, TRAIL Thesis Series, The Netherlands

Soto Y Koelemeijer, G., *On the behaviour of classes of min-max-plus systems*, T2003/6, September 2003, TRAIL Thesis Series, The Netherlands

Lindveld, Ch..D.R., *Dynamic O-D matrix estimation: a behavioural approach*, T2003/7, September 2003, TRAIL Thesis Series, Eburon, The Netherlands

Weerdt, de M.M., *Plan Merging in Multi-Agent Systems,* T2003/8, December 2003, TRAIL Thesis Series, The Netherlands

Langen, de P.W, *The Performance of Seaport Clusters*, T2004/1, January 2004, TRAIL Thesis Series, The Netherlands

Hegyi, A., *Model Predictive Control for Integrating Traffic Control Measures*, T2004/2, February 2004, TRAIL Thesis Series, The Netherlands

Lint, van, J.W.C., *Reliable Travel Time Prediction for Freeways*, T2004/3, June 2004, TRAIL Thesis Series, The Netherlands

Tabibi, M., *Design and Control of Automated Truck Traffic at Motorway Ramps*, T2004/4, July 2004, TRAIL Thesis Series, The Netherlands

Verduijn, T. M., *Dynamism in Supply Networks: Actor switching in a turbulent business environment*, T2004/5, September 2004, TRAIL Thesis Series, The Netherlands

Daamen, W., *Modelling Passenger Flows in Public Transport Facilities*, T2004/6, September 2004, TRAIL Thesis Series, The Netherlands

Zoeteman, A., *Railway Design and Maintenance from a Life-Cycle Cost Perspective: A Decision-Support Approach,* T2004/7, November 2004, TRAIL Thesis Series, The Netherlands

Bos, D.M., *Changing Seats: A Behavioural Analysis of P&R Use,* T2004/8, November 2004, TRAIL Thesis Series, The Netherlands

Versteegt, C., *Holonic Control For Large Scale Automated Logistic Systems,* T2004/9, December 2004, TRAIL Thesis Series, The Netherlands

Wees, K.A.P.C. van, *Intelligente voertuigen, veiligheidsregulering en aansprakelijkheid. Een onderzoek naar juridische aspecten van Advanced Driver Assistance Systems in het wegverkeer,* T2004/10, December 2004, TRAIL Thesis Series, The Netherlands

Tampère, C.M.J., *Human-Kinetic Multiclass Traffic Flow Theory and Modelling: With Application to Advanced Driver Assistance Systems in Congestion,* T2004/11, December 2004, TRAIL Thesis Series, The Netherlands

Rooij, R.M., *The Mobile City. The planning and design of the Network City from a mobility point of view*, T2005/1, February 2005, TRAIL Thesis Series, The Netherlands

Le-Anh, T., *Intelligent Control of Vehicle-Based Internal Transport Systems*, T2005/2, April 2005, TRAIL Thesis Series, The Netherlands

Zuidgeest, M.H.P., *Sustainable Urban Transport Development: a Dynamic Optimization Approach*, T2005/3, April 2005, TRAIL Thesis Series, The Netherlands

Hoogendoorn-Lanser, S., *Modelling Travel Behaviour in Multimodal Networks*, T2005/4, May 2005, TRAIL Thesis Series, The Netherlands

Dekker, S., *Port Investment – Towards an integrated planning of port capacity*, T2005/5, June 2005, TRAIL Thesis Series, The Netherlands

Koolstra, K., *Transport Infrastructure Slot Allocation*, T2005/6, June 2005, TRAIL Thesis Series, The Netherlands

Vromans, M., *Reliability of Railway Systems*, T2005/7, July 2005, TRAIL Thesis Series, The Netherlands

Oosten, W., *Ruimte voor een democratische rechtsstaat. Geschakelde sturing bij ruimtelijke investeringen*, T2005/8, September 2005, TRAIL Thesis Series, Sociotext, The Netherlands

Le-Duc, T., *Design and control of efficient order picking*, T2005/9, September 2005, TRAIL Thesis Series, The Netherlands

Goverde, R., *Punctuality of Railway Operations and Timetable Stability Analysis*, T2005/10, October 2005, TRAIL Thesis Series, The Netherlands

Kager, R.M., *Design and implementation of a method for the synthesis of travel diary data*, T2005/11, October 2005, TRAIL Thesis Series, The Netherlands

Boer, C., *Distributed Simulation in Industry*, T2005/12, October 2005, TRAIL Thesis Series, The Netherlands

Pielage, B.A., *Conceptual Design of Automated Freight Transport Systems*, T2005/14, November 2005, TRAIL Thesis Series, The Netherlands

Groothedde, B., *Collaborative Logistics and Transportation Networks, a modeling approach to network design*, T2005/15, November 2005, TRAIL Thesis Series, The Netherlands

Valk, J.M., *Coordination among Autonomous Planners*, T2005/16, December 2005, TRAIL Thesis Series, The Netherlands

Krogt, R.P.J. van der, *Plan Repair in Single-Agent and Multi-Agent Systems*, T2005/17, December 2005, TRAIL Thesis Series, The Netherlands

Bontekoning, Y.M., *Hub exchange operations in intermodal hub-and-spoke networks. A performance comparison of four types of rail-rail exchange facilities,*T2006/1, February 2006, TRAIL Thesis Series, The Netherlands

Lentink, R., *Algorithmic Decision Support for Shunt Planning,* T2006/2, February 2006, TRAIL Thesis Series, The Netherlands

Ngoduy, D., *Macroscopic Discontinuity Modeling for Multiclass Multilane Traffic Flow Operations,* T2006/3, April 2006, TRAIL Thesis Series, The Netherlands

Vanderschuren, M.J.W.A., *Intelligent Transport Systems for South Africa. Impact assessment through microscopic simulation in the South African context,* T2006/4, August 2006, TRAIL Thesis Series, The Netherlands

Ongkittikul, S., *Innovation and Regulatory Reform in Public Transport,* T2006/5, September 2006, TRAIL Thesis Series, The Netherlands

Yuan, J., *Stochastic Modelling of Train Delays and Delay Propagation in Stations,* T2006/6, October 2006, TRAIL Thesis Series, The Netherlands

Viti, F., *The Dynamics and the Uncertainty of Delays at Signals*, T2006/7, November 2006, TRAIL Thesis Series, The Netherlands

Huisken, G., *Inter-Urban Short-Term Traffic Congestion Prediction*, T2006/8, December 2006, TRAIL Thesis Series, The Netherlands

Feijter, R. de, *Controlling High Speed Automated Transport Network Operations*, T2006/9, December 2006, TRAIL Thesis Series, The Netherlands

Makoriwa, C., *Performance of Traffic Networks. A mosaic of measures*, T2006/10, December 2006, TRAIL Thesis Series, The Netherlands

Miska, M., *Microscopic Online Simulation for Real time Traffic Management*, T2007/1, January 2007, TRAIL Thesis Series, The Netherlands