A High Resolution Cellular Automata Traffic Simulation Model with Application in a Freeway Traffic Information System

Sigurður F. Hafstein[1], Roland Chrobok[1], Andreas Pottmeier[1],
Joachim Wahle[2], and Michael Schreckenberg[1]
[1]Physics of Transport and Traffic, University of Duisburg-Essen,
Lotharstraße 1, 47048 Duisburg, Germany
email: {hafstein,chrobok,pottmeier,schreckenberg}@traffic.uni-duisburg.de
[2]TraffGo GmbH, Falkstraße 73-77, 47058 Duisburg, Germany
email: wahle@traffgo.com

Abstract. *In this article we describe a novel traffic information system for the freeway traffic in North Rhine-Westphalia, the most populous German state. It consists of more than 4,000 loop detectors, a simulator, and a microscopic and a macroscopic graphical interface. These should be considered as 'data input', 'data processing', and 'data output' respectively. First, we discuss the loop detectors. Their mode of operation, how and where they are located, and the quality of their measurements. Next, we describe the simulator, especially its high resolution cellular automaton model of traffic flows, the abstraction of the road network into tracks and nodes, how the data from the loop detectors is integrated, and we give some details on an efficient implementation of the dynamics. Finally, we discuss the graphical interfaces, which display the simulated traffic states, and we give some concluding remarks. In particular, we present the traffic information web page www.autobahn.nrw.de, where the simulated actual traffic state on the freeway network in North Rhine-Westphalia can be sighted.*

1 Introduction

Efficient vehicular transport of people and goods is of vital importance to any modern society. In densely populated areas the capacity of the road network is often at its limits and frequent traffic jams and congestions cause a significant economic damage. Moreover, in these areas, it is usually hardly possible or socially untenable to build more roads. An intelligent use of the resource 'traffic infrastructure' is therefore economically crucial. The German state North Rhine-Westphalia (NRW) is an example of such a densely populated area, where the capacity of the road network is not able to satisfy the traffic demand during the rush-hours. Every day there are congestions on the autobahns (the German freeways) in the Rhine-Ruhr region (Dortmund, Duisburg, Düsseldorf, Essen, etc.) and in the area around Cologne and Leverkusen. To make things even worse, the traffic demand is still growing. For this reason, reliable traffic information systems and traffic management concepts are needed.

The cornerstone of every information system is data. In order to gather data regarding the traffic on the autobahns the German federal Ministry of Transport, Building and Housing equips the autobahns with devices that measure traffic data characteristics. The state of the art devices for such measurements are the so-called *loop detectors*, which are described in more detail in the next section. Loop detectors are locally installed and can only make measurements on the vehicles that drive over them. This implies that a closely meshed series of loop detectors is needed to deliver accurate information on the traffic state of a freeway. Because the installation and maintenance of loop detectors is expensive, it is more economical to install less loop detectors and then use an intelligent method to derive the traffic state in less densely equipped areas. Our approach is to use the loop detector data as a feedback to control the traffic state of a microscopic traffic simulator, which uses a high resolution cellular automaton model of freeway traffic.

In 1992 Nagel and Schreckenberg proposed a stochastic cellular automaton model of vehicular traffic [17], which was able to reproduce some empirically observed non-trivial traffic phenomena like spontaneous traffic jam formation. This publication captured the interest of the physicists community and ever since there has been a continuous progress in the development of cellular automata models of vehicular traffic. The most recent models are able to reproduce free flow, spontaneous jam formation, synchronized traffic, and meta-stability. In Section 3 we describe the cellular automaton model we use to simulate the traffic on the autobahn network in NRW. Further, we consider some algorithmic implementation details and discuss some of the challenges that arise when using this model to simulate the traffic on such a huge and topologically complex road network. This last point is of great importance because the traffic model we use, like most or all traffic models, was developed and tested on topologically simple road networks and the translation to large and topologically complex real road networks, like the autobahn network of NRW, is non-trivial.

In Section 4 we describe the graphical interfaces we have developed to visualize the traffic states generated by the simulator. In order to see what is really happening in the simulator we have a microscopic graphical interface, in which all vehicles on an arbitrary autobahn segment are drawn. Nowadays, powerful 3D graphics accelerators are a commodity. For this reason we implemented it in OpenGL as a true 3D world, in which the viewing angle and the viewing position can be freely chosen. The performance hit when run on top of the simulator is hardly noticeable. The advantage of using 3D graphics is that it looks realistic so potential artifacts in the modeling or bugs in the dynamic part of the simulator code can easily be identified. The microscopic graphical interface can be viewed in Figure 1.

The microscopic graphical interface suits its purpose very well, but is not practical when the macroscopic traffic state is of interest. Therefore, we additionally implemented a macroscopic graphical interface. In this interface the autobahn network is drawn on a map of NRW and the autobahns are partitioned in tracks. Every track is then colored according to the current simulated traffic state.

2 Data Input - Loop Detectors

Data regarding the traffic on the autobahns in NRW are mainly provided by more than 4,000 loop detectors. Loop detectors, or more exactly inductive loop detectors, have become the most common vehicle detection systems since their introduction in the early 1960's. Inductance is a circuit element, usually a conducting coil, in which electrical current is generated by electro-magnetic flux through the circuit. The principal components of an inductive loop detector system include one or more turn of loop wire wound in a shallow slot sawed in the pavement, a lead-in cable to a gate operator cabinet, and an electronic detector unit housed in the cabinet. An ordinary detector unit drives altering current through the loop system at frequencies in the range of about 10 kHz to 200 kHz. The loop system forms a tuned electrical circuit of which the loop wire is the inductive element. Every time a vehicle passes over or is stopped within the loop it decreases the inductance of the loop. This decrease in inductance then actuates the detector output relay (or circuit) which, in turn, sends an impulse to the controller unit signifying that it has detected the passage or presence of a vehicle. Inductive loop detectors sense metal surfaces, not, as often believed, heavy metal mass. The ferrous heavy metal engine in the loop area increases the inductance because of the ferromagnetic effect, but the peripheral metal has the opposite effect and more than offsets the increase from the mass of the engine and the net effect is an overall reduction in inductance. By installing a detector consisting of two inductive loops with a short (known) distance between them, it can measure the speed of a passing vehicle. Because the occupancy, i.e., the time a vehicle is within an inductive loop, can also be measured, such a double loop detector can additionally measure the lengths of passing vehicles.

The loop detectors that are installed on the autobahn in NRW are subject to the technical delivery conditions as laid down in [3]. Two double loop detector geometries are designated, short and long. The exact geometries can be extracted from Figure 2. The loop detectors on the autobahns in NRW deliver aggregated measured data to central servers every minute. The most important measured quantities include:

J_{veh} Total number of vehicles that passed the loop detector in the last minute.

J_{tru} Total number of trucks that passed the loop detector in the last minute.

v_{pac} The average velocity of the passenger cars that passed the loop detector in the last minute.

v_{tru} The average velocity of the trucks that passed the loop detector in the last minute.

The target accuracy of these data laid down by [3] can be sighted in the following table.

Quantity	Case	Tolerance
J_{veh}	≤ 10 vehicles/min	$< 20\%$
J_{veh}	> 10 vehicles/min	$< 10\%$
J_{tru}	≤ 10 vehicles/min	$< 35\%$
J_{tru}	> 10 vehicles/min	$< 20\%$
v_{pac}	≤ 100 km/h	< 3 km/h
v_{pac}	> 100 km/h	$< 3\%$

Further, it is demanded that the fault tolerance of J_{veh} is less than 3% and that of J_{tru} is less than 5% when aggregated over one hour. A typical flow J_{veh} measured by a single inductive loop detector on the autobahn A1 on a Friday and on a Sunday is shown in Figure 3. Note the large fluctuations in the measured flow.

The quantities we use for the feed back control of the simulator are the passenger car flow $J_{pac} := J_{veh} - J_{tru}$, the truck flow J_{tru}, and the vehicle density (vehicles per km)

$$\rho := \frac{J_{veh}^2}{J_{pac} v_{pac} + J_{tru} v_{tru}}. \quad (1)$$

It should be noted that relation (1) for the density ρ is an approximation. It is only exact when all passenger cars pass the loop detector with the velocity v_{pac} and all trucks with the velocity v_{tru}. The true relation for the temporally averaged density over the loop detector is

$$\rho = \sum_i \frac{1}{v_i} [\text{min}],$$

where the index i represents the vehicles that pass the detector and v_i is the velocity of the i-th vehicle. However, because the loop detectors deliver aggregated data, the relation (1) is used. For an overview of the different estimations of the density see, e.g., [9].

3 Data Processing - Simulation

Our approach to generate the traffic state in the whole autobahn network in NRW from the locally measured traffic characteristics by the loop detectors is to feed the data into a high resolution cellular automaton traffic simulator. The simulator does not only deliver information about the traffic states in regions not covered by measurement, but also delivers reasonable estimates for other valuable quantities like travel times, a quantity that is not directly accessible from the measurements of the detectors. In this section we will first describe the microscopic traffic dynamics model used by the simulator. Then we discuss some details of the data structures and algorithms we use for an efficient implementation of the simulator. Finally, we demonstrate some additional rules that have to be applied when simulating a real complex freeway network.

3.1 Simulation Model

Because data is fed real-time into the simulator, it has to be efficient, that is, at least real time. Due to their design cellular automata models are very efficient in large-scale network simulations [5, 19, 10, 20, 22]. The first cellular automaton model for traffic flow that was able to reproduce some characteristics of real traffic, like jam formation, was suggested by Nagel and Schreckenberg [17] in 1992. Their model has been continuously refined in the last 10 years. The model we implemented in our simulator uses smaller cells in comparison with the original Nagel-Schreckenberg model, a *slow-to-start rule*, *anticipation*, and *brake lights*. With these extensions the cellular automaton traffic model is able to reproduce all empirically observed traffic states. For the interpretation of the different traffic phases in the fundamental diagram of freeway traffic (flow in relation to density) see Figure 4. Further, we use two classes of different vehicles, *passenger cars* and *trucks*, where the trucks have a lower maximum velocity and are not allowed to drive on the left-most lane.

Smaller cells allow a more realistic acceleration and more speed bins. We are currently using a cell size of 1.5 m. Because the time-steps in the simulation model are set to be 1 second, this corresponds to speed bins of 5.4 km/h and an acceleration of 1.5 m/s^2 ($0 - 100$ km/h in 19 s), in comparison to 7.5 m cells and an acceleration of 7.5 m/s^2 ($0 - 100$ km/h in 3.8 s), in the original Nagel-Schreckenberg model. A passenger car occupies $2 - 5$ contiguous cells and a truck occupies 5 contiguous cells. By using a slow-to-start rule [1] meta-stable traffic flows are modeled by the simulator, a phenomenon observed in empirical studies of real traffic flows [7, 11, 24]. By including anticipation and brake lights [2, 13] in the modeling, the vehicles do not solely determine their velocity in dependency of the distance to the next vehicle in front, but also consider the speed and the deceleration of the front vehicle.

In the Nagel-Schreckenberg model there is only one global parameter, the probability constant (or dawdling parameter) p. Further, the dynamical variables of the model are dimensionless, i.e., lengths and positions are expressed in terms of number of cells, velocities are in terms of number of cells per second, and times are in terms of number of seconds. Every vehicle, say vehicle n, is completely determined by two parameters. Its position $x_n(t)$ and its velocity $v_n(t)$ at time t. When the vehicle n decides in the time-step $t \mapsto t+1$ how fast it should drive, it does this by considering the distance $d_{n,m}(t)$, i.e., the number of empty cells, to the next vehicle m in front. The modifications of the Nagel-Schreckenberg model mentioned above, imply that we have to add some new parameters to the model. When the simulation algorithm decides, whether a vehicle n should brake or not, it does not only consider the distance to the next vehicle m in front, but estimates how far the vehicle m will move during this time-step (anticipation). Note, that the moves are done in parallel, so the model remains free of collision. This leads to the effective gap

$$d_{n,m}^{\text{eff}}(t) := d_{n,m}(t) + \max(v_m^{\min}(t) - d_S, 0)$$

seen by vehicle n at time t. In this formula $d_{n,m}(t)$ is the number of free cells between the front of the vehicle n and the back of the vehicle m, d_S is a safety distance, set equal to 6 cells (9 m) in our model, and

$$v_m^{\min}(t) := \min(d_{m,l}(t), v_m(t)) - 1,$$

where $d_{m,l}(t)$ is the number of free cells between the vehicle m and its next vehicle in front l, is a lower bound of how far the vehicle m will move during this time-step.

Brake lights are a further component of the anticipated driving. They allow vehicles to react to disturbances in front earlier by adjusting their speed. Empirical observations suggest [6, 16] that drivers react in a temporal- rather than a spatial-horizon. For this reason the velocity-dependent temporal interaction horizon

$$t_n^S(t) := \min(v_n(t), h)$$

is introduced to the model. The constant h determines the temporal range of interaction with the brake light $b_m(t) \in \{\text{on}, \text{off}\}$ of the next vehicle m in front. The vehicle n does only react to $b_m(t)$ if the time to reach the back of the vehicle m, assuming constant velocity ($v_n = \textit{const.}$) and that the vehicle m stands still, is less than $t_n^S(t)$, that is,

$$t_n^h(t) := \frac{d_{n,m}(t)}{v_n(t)} < t_n^S(t).$$

In our model we take h equal to 7 s.

The third modification of the Nagel-Schreckenberg model implemented in the simulator is a velocity dependent randomization. It means that the probability constant p is replaced with a probability function dependent on the velocity of the vehicle. Further, the probability is also a function of the brake light of the next vehicle in front. In every time-step for every vehicle n with vehicle m next in front, the probability that the vehicle n brakes is

$$p = p(v_n(t), b_m(t)) := \begin{cases} p_b, & \text{if } b_m(t) = \text{on} \\ & \text{and } t_n^h(t) < t_n^S(t), \\ p_0, & \text{if } v_n(t) = 0, \\ p_d, & \text{default.} \end{cases}$$

In our model we take p_b equal to 0.96, p_0 equal to 0.5, and p_d equal to 0.1. This formula for p, where p_0 is significantly larger than p_d, is the so-called slow-to-start rule. It is needed for the empirically observed backward propagation of traffic jam fronts and leads to meta stable traffic flows. The constant p_b controls the upstream propagation of brake lights and is responsible for the correct mapping of synchronized flows and the parameter p_d is determines the degree of fluctuations and thus determines the maximum flow and is the origin of spontaneous jam formation. For a comparison of the fundamental diagram from the model used by the simulator and from empirical data see Figure 5. A detailed comparison of the fundamental diagrams, the time headway distributions, the lane usage, and the autocorrelations and crosscorrelations of density, velocity, and flow in the model presented here with empirical data and earlier models would significantly extend the scope of this paper. For a thorough discussion on these we refer to [14].

To sum up, to move the vehicles forward in the network, the algorithm executes the following steps in parallel for all vehicles n, i.e., every step is executed for all vehicles in the network before moving to the next step:

- Step 0: Initialization:

  For vehicle n find next vehicle in front m.

  Set $p := p(v_n(t), b_m(t))$ and $b_n(t+1) :=$ off.

- Step 1: Acceleration:

$$v_n(t + \frac{1}{3}) := \begin{cases} v_n(t), & \text{if } b_n(t) = \text{on or} \\ & (b_m(t) = \text{on and} \\ & t_n^h(t) < t_n^S(t)), \\ \min(v_n(t){+}1, v_{\max}), & \text{default.} \end{cases}$$

- Step 2: Braking:

$$v_n(t + \frac{2}{3}) := \min(v_n(t + \frac{1}{3}), d_{n,m}^{\text{eff}}(t)).$$

  Turn brake light on if appropriate:

$$\text{if } v_n(t + \frac{2}{3}) < v_n(t), \text{ then } b_n(t+1) := \text{on}.$$

- Step 3: Randomization with probability p:

$$v_n(t{+}1) := \begin{cases} \max(v_n(t + \frac{2}{3}) - 1, 0), & \text{with probability } p, \\ v_n(t + \frac{2}{3}), & \text{default.} \end{cases}$$

  Turn brake light on if appropriate:

$$\text{if } p = p_b \text{ and } v_n(t+1) < v_n(t + \frac{2}{3}),$$
$$\text{then } b_n(t+1) := \text{on}.$$

- Step 4: Move (drive):

$$x_n(t+1) := x_n(t) + v_n(t+1).$$

Free lane changes are needed so that vehicles can pass slower driving passenger cars and trucks. When designing rules for the free lane changes, one should take care of that vehicles passing other vehicles do not disturb the traffic on the lane that they use to pass to much. Further, one has to take account of German laws, which ban passing a vehicle to the left. Further, it is advantageous to prohibit trucks to drive on the leftmost lane in the simulation, because a truck passing another truck forces all vehicles on the left lane to reduce their velocity and produces a deadlock that may not resolve for a long time [12].

One more variable, $l_n \in \{\text{left}, \text{right}, \text{straight}\}$, is needed for the free lane changes. The variable l_n notes if the vehicle n should change the lane during the actual time-step or not. This variable is not needed if the lane changes are executed sequentially, but we prefer a parallel update of the lane changes for all vehicles and that renders this variable necessary. The asymmetric lane changing rules we implemented in the simulator follow those of [14]. They lead to the empirically observed *lane usage inversion* on the German autobahn [15, 23]. Although there is a right lane preference, the distribution of the flow becomes asymmetric and the flow is higher on the left than on the right lane. The lane usage as a function of density in the model used by the simulator can be sighted in Figure 6. For the free lane changes to the left the simulator executes the following steps parallel for all vehicles n:

Passing on the lane to the left:

- Step 0: Initialization:
  For vehicle n find next vehicle in front m on the same lane, next vehicle in front s on the lane left to vehicle n and the next vehicle r behind vehicle s. Set $l_n :=$ straight.

- Step 1: Check lane change:

  if $b_n(t) = \text{off}$ and $d_{n,m}(t) < v_n(t)$

  and $d^{\text{eff}}_{n,s}(t) \geq v_n(t)$ and $d_{r,n}(t) \geq v_r(t)$,

  then set $l_n :=$ left.

- Step 2: Do lane change:

  if $l_n =$ left, then change lane for vehicle n

  to the left.

The definition of the gaps $d^{\text{eff}}_{n,s}(t)$ and $d_{r,n}(t)$ is an obvious extensions of the definition above, one simply considers a copy of the vehicle n on its left side. These passing rules used by the simulator can verbally be summed up as follows: First, a vehicle checks if it is hindered by the predecessor on its own lane. Then it has to take into account the gap to the successor and to the predecessor on the lane to the left. If the gaps allow a safe change the vehicle moves to the left lane. For the right free lane changes the simulator executes the following steps parallel for all vehicles n:

Return to a lane on the right:

- Step 0: Initialization:
  For vehicle n find next vehicle in front s on the lane right to vehicle n and next vehicle r behind vehicle s. Set $l_n :=$ straight.

- Step 1: Check lane change:

  if $b_n(t) = \text{off}$ and $t^h_{n,s}(t) > 3$ and

  $(t^h_{n,m}(t) > 6$ or $v_n(t) > d_{n,m}(t))$

  and $d_{r,n}(t) > v_r(t)$,

  then set $l_n :=$ right.

- Step 2: Do lane change:

  if $l_n =$ right, then change lane for vehicle n

  to the right.

Thus, a vehicle returns to the right lane if there is no disadvantage in regard to its velocity and it does not hinder any other vehicle by doing so.

It should be noted, that it is not possible to first check for all lane changes to the left and to the right and then perform them all in parallel without doing collision detection and resolution. This would be necessary because there are autobahns with three lanes and more. To overcome this difficulty, the lane changes to the left, that is passing, are given a higher priority than the lane changes to the right. For a systematic approach to multi-lane traffic, i.e., lane-changing rules, see, for example, [18]. For a detailed discussion of the different models see [8, 21, 4] and the references therein.

3.2 Some Details on an Efficient Implementation

A crucial point in the design of a large scale traffic simulator is the software-engineering part. On the one hand, the chosen data structures have to be abstract enough to model the occurrences in the whole autobahn network and it should be reasonably easy to generalize and make changes in the design. The latter is particulary of great importance, because there is a continuous progress in the theory of traffic flows and traffic flow modeling and the desire for some extensions in the simulator in the future is a certainty. On the other hand, an efficient algorithmic implementation of the dynamics is a necessity for such a large road network. A simulator that is not able to simulate the traffic flow in at least real time during the rush hours is of little value and if it is intended to use the simulator for traffic forecast, it has to be multiple real time.

Let us start with the representation of the road network in the simulator. Like in other simulators (e.g., [5, 26]) the network consists of basic elements, tracks and nodes. A track is a directed bundle of parallel lanes or, more casually, simply a piece of autobahn. A vehicle on a track has local coordinates (cell and lane) with respect to the track. A node is a connection between two tracks. It stores information about where the position of the exit is on the track to be left, about how to leave the track (lane change, drive out of it), and how to calculate the new local coordinates on the target track (cell offset, lane offset). Every track contains pointers to all nodes relevant to it. Further, a track keeps information in every cell relevant to the vehicles. The vehicles have a fast access to this information through their position (cell and lane). A track stores the vehicles on it in a doubly linked list of pointers to the vehicles, see Figure 7. This list is sorted with regard to the cell positions of the vehicles in every time-step of the simulation. Note, that this is necessary, because the relative order of the vehicles can change due to vehicles passing. By doing this, the vehicles have a fast access to their neighbors, which is essential for the efficiency of the simulator.

By combining tracks and nodes, one is able to build the complex structures of the autobahn network. Examples for these structures are:

- junctions, where vehicles enter or leave the autobahn,
- intersections, at which two autobahns are connected, and,
- triangular intersections, where two autobahns meet,

but one ends.

Other geometries are rarely found in the autobahn network in NRW. The complexity of an intersection can be derived from Figure 8. However, they can be constructed easily with the elements used here. Using these tracks and nodes the autobahn network of NRW was reconstructed. It comprises 3,988 links, 830 on- and off-ramps, and 67 intersections. The overall length of the lanes is approximately 12,200 km, corresponding to more than 8 million 1.5 m cells. The data used for the network were extracted from the *NW*-SIB, a Geographic Information System (GIS) database provided by the state of NRW.

The concept of a vehicle is implemented as a C++ class. The vehicle contains its position (cell and lane), the node it is heading to, a pointer to the track it is on, and various other bookkeeping data. To keep the simulator flexible the most important functions, `CheckLaneChange` and `CalculateVelocity` (see last section), are implemented as static function pointers. An alternative would be to use class inheritance, but this would imply the overhead of virtual functions (every vehicle would carry a pointer to a virtual table where the addresses of the functions to be used is stored), so we decided against it. For efficiency reasons it is crucial to use a custom memory allocation for the vehicles, i.e., redefine operator *new* and *delete* for the vehicles class.

The simulation performs multiple real-time on a modern personal computer and in regard to the ever growing computational power it looks promising to combine the simulation with historical data for a traffic forecast. We are currently working on this and the preliminary results look promising. A further application for the simulator is to research the influence of new roads or construction areas on the global traffic flow.

3.3 Additional Rules for Real Traffic Simulation

The cellular automaton model for traffic flow used by the simulator was designed to be able to reproduce the main aspects of the fundamental diagram for real traffic flows (vehicle flow as a function of vehicles per km) and the most important microscopic properties, like the time headway distribution. This ability was verified by testing it on topologically simple networks. When simulating the traffic on a large and topologically complex network, like the autobahn network in NRW, some extensions to the cellular automaton model have to be introduced. One is the guidance of vehicles and another is a strategy to integrate the loop detectors data into the simulation.

A real driver usually has the intention to reach some goal with his driving. This makes it necessary to incorporate routes in the modeling. In principle, there are two different strategies to solve this problem. One can assign an origin and a destination to each vehicle, choose a route between the origin and the destination, and then guide it through the network according to this route [19, 20]. For the autobahn network in NRW origin-destination information with a sufficient temporal and spatial resolution is not available. Therefore, the vehicles are guided in the network according to the probabilities calculated on the basis of the measured data. This means that a vehicle is not guided through the whole network, but every time it reaches a new track it will decide in accordance with the measured probabilities how it leaves the tracks, i.e., which node it chooses.

To implement this we use forced lane changes. Forced lane changes are necessary so that the vehicles can drive from on-ramps onto the autobahn, from the autobahn onto off-ramps, when the autobahn narrows, and when vehicles drive from one particular section of the autobahn onto another over an intersection. Forced lane changes differ from free lane changes in a fundamental way. While free lane changes give vehicles the opportunity to pass slower driving vehicles and thus reduce disturbances, forced lane changes stem from the need to reach a node and are obviously a source for disturbances.

The simulator uses gradually increasing harsh measures to force lane changes. At the beginning of an area where a vehicle could change to the target lane, it does so if the gap is sufficiently large and no vehicle is severely hindered. At the end of the area it will bully into any gap regardless of velocity differences. Further, a vehicle driving on its target lane should not leave the lane to pass another vehicle. An efficient implementation of this strategy is to store the lane change information in the cells. This gives a fast access through the coordinates of a vehicle. Of course this information depends on the node chosen and whether the vehicle is a truck or a passenger car. Because of this, every track has several versions of the lane change information.

To incorporate the real world measurements from the loop detectors into the simulation, vehicle-moving, inserting, and removing algorithms have to be applied. This is done at the so-called checkpoints, which are located at those places in the network where a complete cross-section is available, i.e., all lanes are equipped with a loop detector. Every time, when checkpoint-data is provided, the simulator uses the measured values to adjust the traffic state in the simulation. The first step is to try to move vehicles behind the checkpoint in front of it and vice versa. If this is not enough to adjust the traffic state, vehicles are inserted or removed. This should be preferred to pure insert/removal strategies, because these can completely fail due to positive feedback if a non-existing traffic jam is produced by the simulator. In this case the simulator measures a much lower flow than the loop detectors, so vehicles are added periodically to the ever growing traffic jam leading to a total breakdown.

For realistic results it is further important to minimize the perturbation of the dynamics present in the network due to the data integration. Therefore, we propose a method which follows the idea to add the vehicles to the network "adiabatically", i.e., without disturbing the system. If the number of vehicles crossing the checkpoint is lower than measured by the detector, vehicles are inserted with regard to the measured mean velocity and mean gap, so that the system is not disturbed, i.e., no vehicle has to brake due to the insertion. This method is therefore called "Tuning of the Mean Gap" [10].

4 Data Output - Visualization of the Simulated Traffic

As mentioned in the introduction we have two different graphical interfaces to the simulated traffic, microscopic and macroscopic. In the microscopic graphical interface the individual vehicles are drawn in 3D and in the macroscopic the tracks are colored according to their current traffic state. The microscopic graphical interface was implemented for internal use. It is a valuable tool to verify the (visual) correctness of the dynamics. It is possible to choose an arbitrary position in the autobahn network as a viewpoint or to follow an individual vehicle including all its dynamical variables. It was implemented in OpenGL using the open source *OpenGL Utility Toolkit (GLUT)*, which is a system independent window programming toolkit for 3D graphics. Although the microscopic graphical interface is very useful for its purpose, it is not dedicated to visualize the macroscopic traffic state in a large area.

The design of the simulator was financially supported by the Ministry of Transport, Energy and Spatial Planning of North Rhine - Westphalia, the reason being, that it wanted a novel web-based traffic information system for the public. This information system is provided by a Java applet at the URL *http://www.autobahn.nrw.de*, which was implemented in a collaboration with TraffGo GmbH. The Java applet draws a map of NRW, where the autobahns are colored according to the simulated traffic state, from light green for free flow, over dark green and dark yellow for dense and very dense synchronized flow, to red for a traffic jam. In Figure 9 the macroscopic traffic state as displayed by the applet can be sighted. In Figure 10 the connection between the microscopic and macroscopic graphical interfaces is shown.

The Java applet displays some other information of use to the drivers, the most important being construction areas and lane blockages. Construction areas are drawn at the appropriate positions on the map and their estimated influence on the traffic is shown through red construction signs for a high risk of a traffic jam and green construction signs for a low risk. Lane blockages are drawn at the appropriate positions and their cause is written in a window at the bottom. For both construction areas and lane blockades, the estimated duration is available too. The response to this novel information system has been very positive and TV-stations, newspapers, and magazines have made positive tests where they compare the actual traffic state to the traffic state presented by our simulator. At the moment we are working on an extended information system, where not only the actual traffic state is graphically available, but also a prognosis for the traffic state in 30 minutes.

5 Summary

In this paper we present a novel traffic information system for the autobahn network in North Rhine -Westphalia, the most populous German state. This information system consists of three parts: data input, data processing, and data output. The data input are the measurements of more than 4,000 loop detectors that are installed locally on the autobahn and deliver data minute by minute to central servers. The data processing is done by an microscopic traffic simulator. The simulator uses an advanced cellular automaton model of traffic flow and adjusts the traffic state in accordance with measurements of the real traffic flow provided by the loop detectors. The cellular automaton model, the abstraction of the network, the guidance of the vehicles, and the data integration strategies to periodically adjust the traffic flow in the simulation in accordance with the measured flow on the autobahn were discussed, as well as some details on an efficient implementation of the dynamics. The data output is provided by a microscopic and a macroscopic graphical interface. The microscopic graphical interface draws the vehicles on an arbitrary piece of the autobahn in 3D. The macroscopic graphical interface colors the autobahn on a map of NRW according to the simulated current traffic state. It is available at www.autobahn.nrw.de.

We are working on several extension of this traffic information system, the most important being the generation of a high quality traffic prognosis based on the actual simulated traffic and historical data. Further, we are extending the simulator so that it can be used for the research of traffic flow control. This is not as simple as it might seem, because any information about the current traffic state available to the public is likely to influence the strategy of the drivers [25].

References

[1] Barlovic R., Santen L., Schadschneider A., Schreckenberg M. (1998) Metastable states in cellular automata for traffic flow. Eur. Phys. J. B **5**, 793–800

[2] Barrett C., Wolinsky M., Olesen M. (2000) Emergent local control properties in particle hopping traffic simulations. In: Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) (2000) Traffic and Granular Flow '99. Springer, Heidelberg

[3] Bundesminister für Verkehr (1993) Technische Lieferbedingungen für Streckenstationen. Verkehrsblattverlag, Dortmund

[4] Chowdhury D., Santen L., Schadschneider A. (2000) Statistical Physics of Vehicular Traffic and Some Related Systems. Physics Reports **329**, 199–329

[5] Esser J., Schreckenberg M. (1997) Microscopic simulation of urban traffic based on cellular automata. Int. J. of Mod. Phys. C **8**, 1025–1036

[6] George H. (1961) Measurement and Evaluation of Traffic Congestion. Bureau of Highway Traffic, Yale University, 43–68

[7] Helbing D. (1996) Empirical traffic data and their implications for traffic modelling. Phys. Rev. E **55**, R25

[8] Helbing D., Herrmann H., Schreckenberg M., Wolf D. (Eds.) (2000) Traffic and Granular Flow '99. Springer, Heidelberg

[9] Helbing D. (2001) Traffic and related self-driven many-particle systems. Rev. of Modern Phys. **73**, 1067–1141

[10] Kaumann O., Froese K., Chrobok R., Wahle J., Neubert L., Schreckenberg M. (2000) On-line simulation of the freeway network of North Rhine-Westphalia. In: [8], 351–356

[11] Kerner B., Rehborn H. (1997) Experimental properties of phase transitions in traffic flow. Phys. Rev. Lett. **79**, 4030-4033

[12] Knospe W., Santen L., Schadschneider A., Schreckenberg M. (1999) Disorder effects in cellular automata for two-lane traffic. Physica A **265**, 614-633

[13] Knospe W., Santen L.,Schadschneider A., Schreckenberg M. (2000) Towards a realistic microscopic description of highway traffic. J. Phys. A **33**, L1–L6

[14] Knospe W. (2002) Synchronized traffic: Microscopic modeling and empirical observations. PhD thesis, Gerhard-Mercator-Universität Duisburg

[15] Leutzbach W., Busch F. (1984) Spurwechselvorgänge auf dreispurigen BAB-Richtungsfahrbahnen. Universität Karlruhe

[16] Miller A. (1961) A queuing model for road traffic flow. J. of the Royal Stat. Soc. **B1**, 23, University Tech. Rep. PB 246, Columbus, USA, 69–75

[17] Nagel K., Schreckenberg M. (1992) A cellular automaton model for freeway traffic. J. Physique I **2**, 2221–2229

[18] Nagel K., Wolf D. E., Wagner P., Simon P. (1998) Two-lane traffic rules for cellular automata: A systematic approach. Phys. Rev. E **58**, 1425–1437

[19] Nagel K., Esser J., Rickert M. (2000) Large-scale traffic simulations for transport planning. In: Stauffer D. (Ed.), Ann. Rev. of Comp. Phys. **VII**, 151–202, World Scientific, Singapore

[20] Rickert M., Wagner P. (1996) Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. Int. J. of Mod. Phys. C **7**, 133–153

[21] Schreckenberg M., Wolf D. (Eds.) (1998) Traffic and Granular Flow ’97. Springer, Singapore

[22] Schreckenberg M., Neubert L., Wahle J. (2001) Simulation of traffic in large road networks. Future Generation Computer Systems, **17**, 649–657

[23] Sparmann U. (1978) Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen. In: Forschung Straßenbau und Straßenverkehrstechnik Heft 263, Bundesminister für Verkehr

[24] Treiterer J. (1975) Investigation of traffic dynamics by areal photogrammatic techniques. Tech. report, Ohio State University Tech. Rep. PB 246, Columbus, USA

[25] Wahle J., Bazzan A., Klügl F., Schreckenberg M. (2000) Anticipatory Traffic Forecast Using Multi-Agent Techniques. In: [8], 87–92

[26] Yang Q., Koutsopoulos H. N. (1996) A microscopic traffic simulator for evaluation of dynamic traffic management systems. Transp. Res. C **4**, 113–129

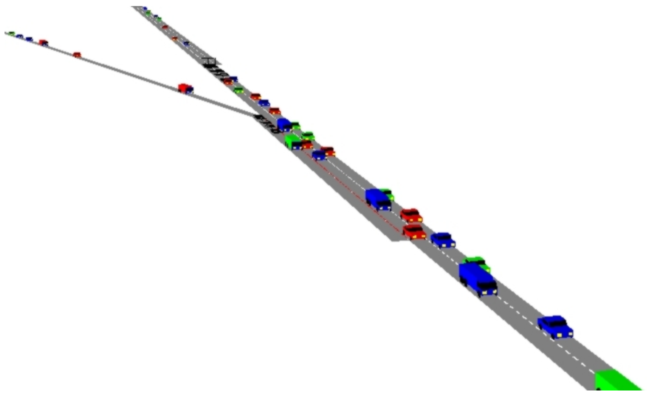


Figure 1: Part of the autobahn in the simulator.

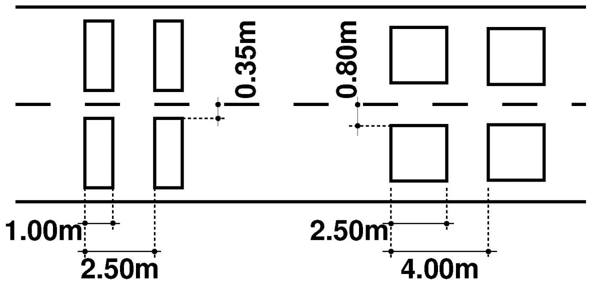


Figure 2: The double inductive loop detector geometries from [3].

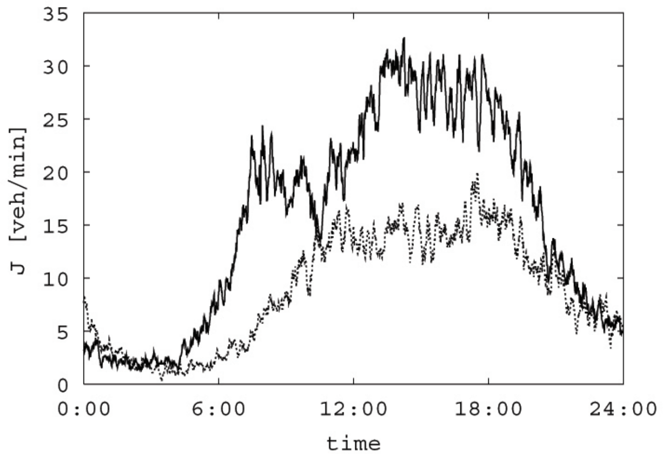


Figure 3: The flow measured by an inductive loop detector on a Friday (the solid line) and on a Sunday (the dashed line).

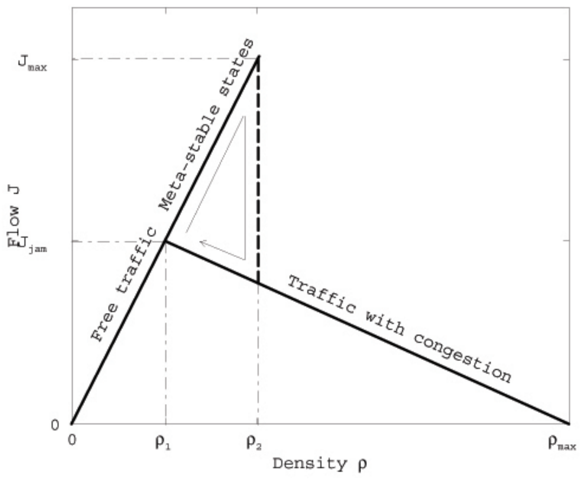


Figure 4: The different traffic states in the fundamental diagram.

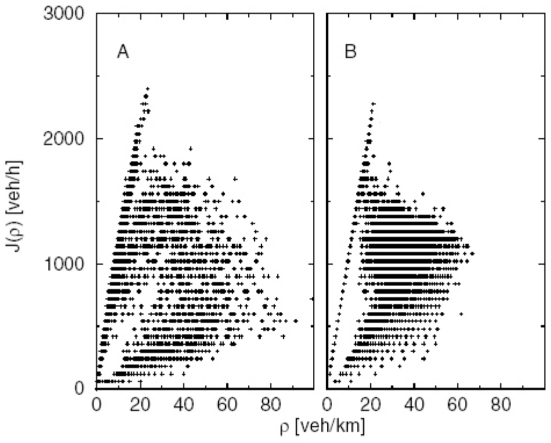


Figure 5: Fundamental diagrams from the traffic model (B) and from empirical data (A).

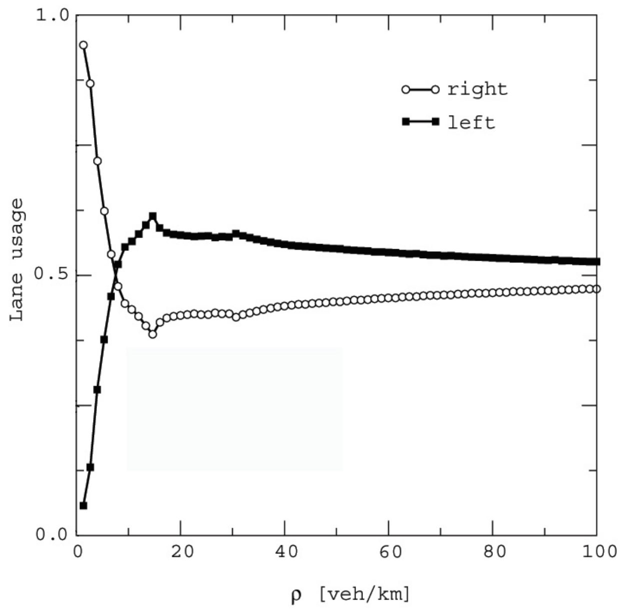


Figure 6: Lane usage as a function of density for the asymmetric lane change rules used by the simulator.

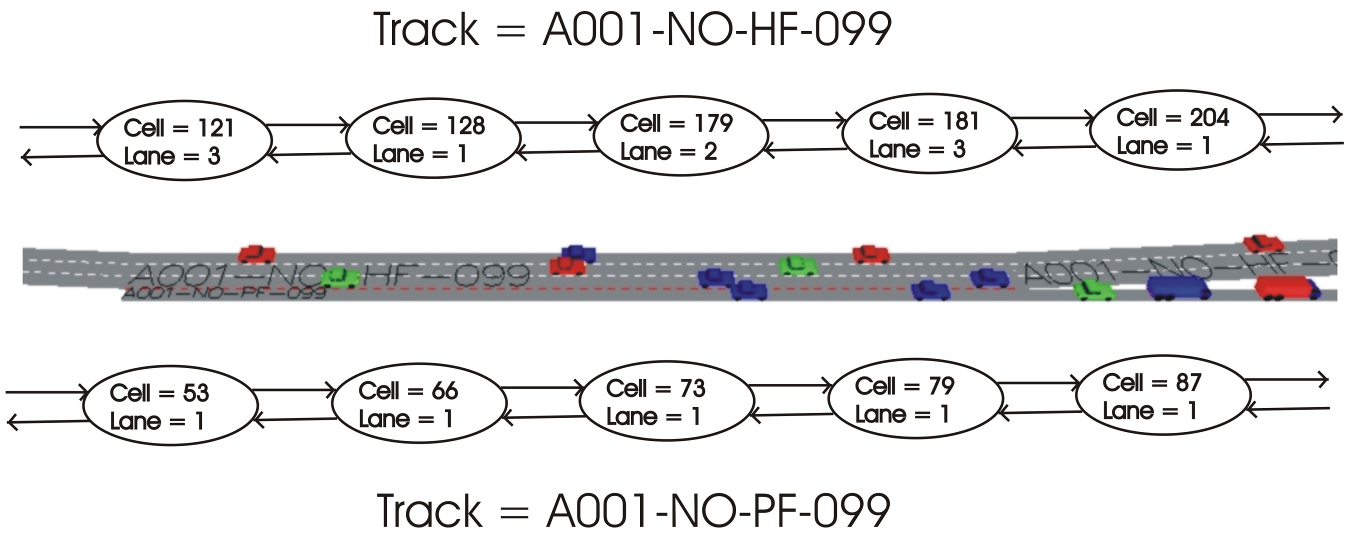


Figure 7: The tracks in the simulator store its vehicles as sorted doubly linked lists.

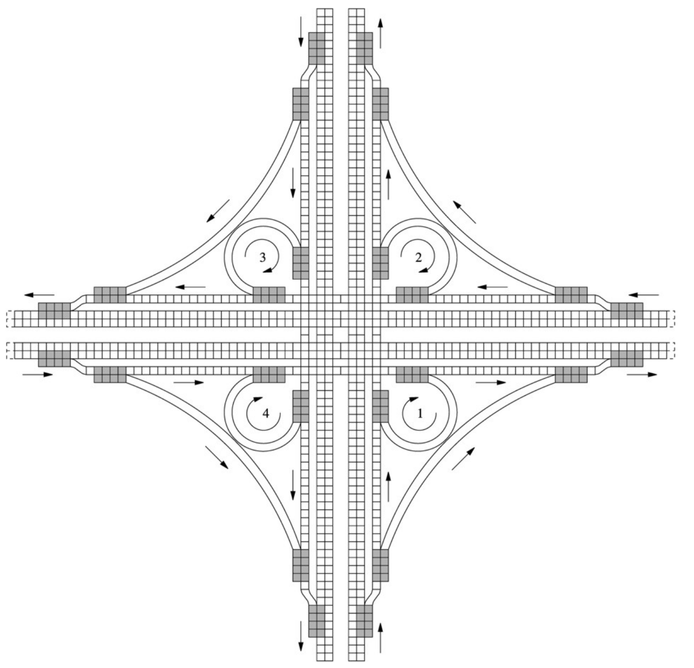
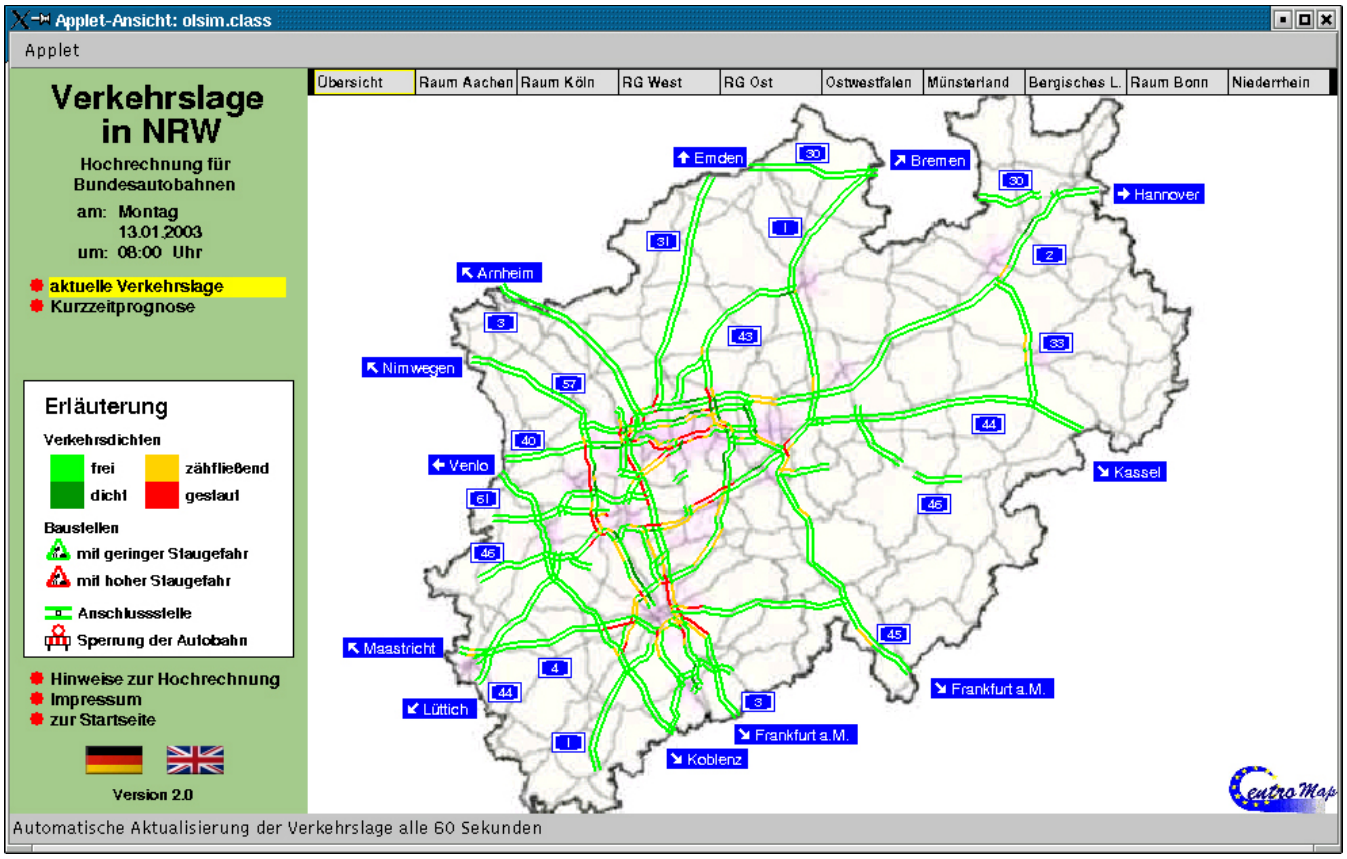


Figure 8: The complex structure of an intersection.



Verkehrslage in NRW

Hochrechnung für Bundesautobahnen

am: Montag 13.01.2003
um: 08:00 Uhr

Erläuterung

Verkehrsdichten

🟩	frei	🟨	zähfließend
🟩	dicht	🟥	gestaut

Baustellen
- ⚠ mit geringer Staugefahr
- ⚠ mit hoher Staugefahr
- Anschlussstelle
- Sperrung der Autobahn

Automatische Aktualisierung der Verkehrslage alle 60 Sekunden

Figure 9: The simulated traffic state on the autobahns in NRW on the 13th January 2003. Due to bad weather conditions all major autobahns were jammed.

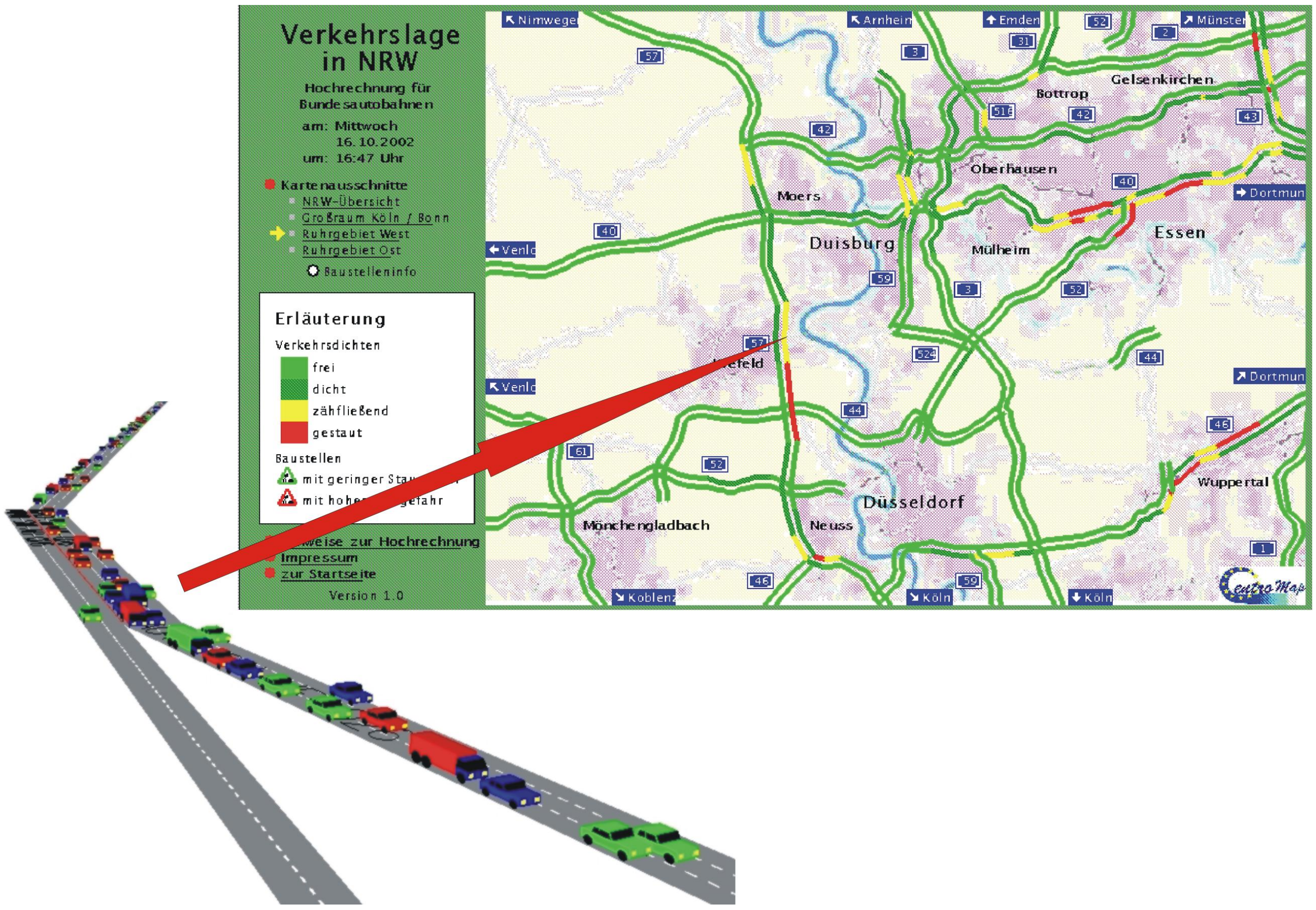


Figure 10: In the Java applet the macroscopic simulated traffic state is presented graphically.