Integrating Simple yet Robust Protocol Layers for Wireless Ad Hoc Intervehicle Communications

Linda Briesemeister and Günter Hommel
Technical University of Berlin
School of Electrical Engineering and Computer Science
Einsteinufer 17
10587 Berlin, Germany
linda.briesemeister@gmx.de, hommel@cs.tu-berlin.de

Keywords Mobile ad hoc network, intervehicle communication, Specification and Description Language (SDL)

Abstract

We present an approach to simulating ad hoc network protocols integrated with their intended application. We implement the protocol layers and the application using the Specification and Description Language (SDL). Then, we simulate the ad hoc network directly from the SDL model. Using the interface of the SDL software tool, we incorporate the mobility model for hosts described in trace files.

We suggest applying mobile ad hoc networks to communicating vehicles in road traffic. As an example, we look at vehicles exchanging messages to detect traffic jams on highways. Simulations of this application in a realistic traffic scenario demonstrate the powerful effect and robustness of rather simple, distributed protocols communicating through a mobile ad hoc network.

Introduction

Many recent studies have proposed and simulated single layers of communication protocols for mobile ad hoc networks [1, 2, 3, 4]. An ad hoc network consists of mobile hosts that communicate via wireless links. Due to mobility, the topology of the network changes continuously and wireless links break down and reestablish frequently. No fixed infrastructure supports the communication in ad hoc networks. Undoubtedly, the mobility of hosts influences the performance of these networks [5].

We believe that studying the application—and hence the mobility model as well—plays a significant role in understanding ad hoc networks. As an example, we focus on applying ad hoc networks to direct wireless communication between vehicles in road traffic [6, 7]. The vehicles are equipped with a computer controlled radio modem allowing them to contact other equipped vehicles in their vicinity. In our application, the vehicles collaborate through such an ad hoc network to detect traffic jams on highways. Tests with off-the-shelf devices [8, 9] have shown the general feasibility of radio modems in the 2.4 GHz industrial, scientific, and medical (ISM) band. In field experiments [8], connections lasted 5 s between two oncoming cars driving at a speed of 130 km/h in opposite directions on a German highway.

The vehicles disseminate messages using scoped flooding in which the outermost receivers forward messages earlier than receivers closer to the sender of the message [10]. As a second modification of simple flooding, we make vehicles wait to forward a message until they find new communication partners [6]. This mechanism helps to cope with network fragmentation which occurs due to sparse deployment of the system when introduced to the market.

In this article, we propose integrating the protocol stack with the intended application and simulating the network in a realistic scenario. We specify our algorithms using the Specification and Description Language (SDL) [11, 12] which exhibits object orientation. Thus, SDL can express multiple instances of the protocol stack forming the network. SDL offers rich semantics allowing the software tools to simulate the behavior of the model. Through the interface of the used SDL tool, we combine the network simulation with a mobility model that we derive from a microscopic traffic simulation.

Proposed Protocols

The intervehicle communication system consists of five layers modeled as SDL blocks. Figure 1 depicts the system architecture for the device used to equip a vehicle. We assume that the system is connected to a navigation system, which inputs data such as the current position, velocity, and street type. In this section, we sketch the algorithms of each block starting with the lowest layer. For a detailed description and implementation of the system refer to [13].

As an interface to the wireless channel, we implement a

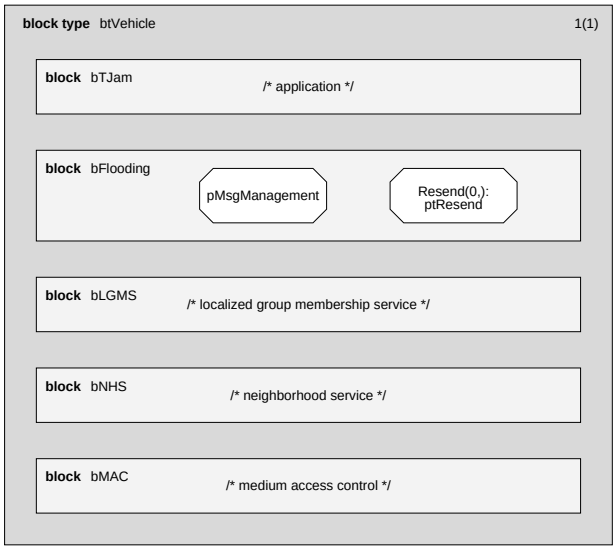


Figure 1. Overview of system architecture

simple medium access control in block *bMAC*. The medium access control offers two primitives to the upper layers—sending and receiving a packet. If a send request reaches the layer while busy either sending or receiving another data packet, the packet requesting to be sent is dropped. If multiple transmissions of packets overlap in time at the receiver, all of them are discarded as collided packets. The block *bMAC* thus resembles a carrier sense multiple access scheme, in which no back-off strategy exists—if the channel is busy, then the packet to be sent is lost. Hence, packets are sent immediately without delay if the channel is idle.

The next layer *bNHS* implements a neighborhood service that keeps track of the list of current neighbors. The block *bNHS* employs a simple heartbeat mechanism in which it sends its own identity periodically like a heartbeat. It also collects the heartbeat messages from other vehicles. When *bNHS* stops receiving the heartbeat of existing neighbors, it removes them from its list. If the neighborhood service detects a heartbeat from unknown neighbors, it adds them to its list. The neighborhood service reports all changes in the neighborhood list to the upper layers.

The localized group membership service *bLGMS* tracks the membership of the adjacent neighbors. The application determines the membership of its own system and requests to join or to leave the group. Then, *bLGMS* installs views on the membership of neighbors. A view comprises the identities of neighbors which are currently perceived as members. Non-members always have an empty view. Members are always included in their own view.

The layer *bFlooding* responsible for disseminating messages through the ad hoc network uses a process *pMsgManagement* and a generic process type *ptResend*. The process *pMsgManagement* keeps track of previously seen messages.

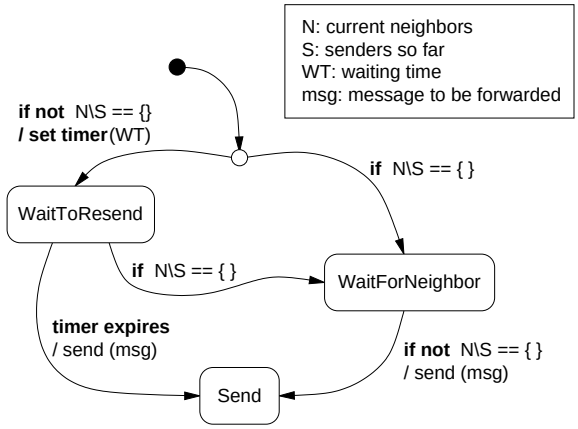


Figure 2. State chart of *ptResend* process instances

The system uses scoped flooding to route messages through the ad hoc network. We limit the propagation of a message to a certain number of hops that a message can take and to a given life time. Every time it forwards a message, the process *pMsgManagement* spawns a new instance of the process type *ptResend*. The state transition diagram for *ptResend* is depicted in Figure 2. We distinguish between two states "WaitToResend" and "WaitForNeighbor."

The process of type *ptResend* initializes a list of known senders with the identity of the message's sender. Then, the process computes the set difference of the current neighbors without the known senders to decide on the next step. If this set difference is not empty, the vehicle has neighbors other than the sender of the previously received message, and the process enters the "WaitToResend" state. A timer is set to a waiting time WT determined by the distance d of the vehicle to the sender of the message. The function for the time to wait WT yields a shorter value for more distant senders, such as given in Equation 1:

$$WT(d) = -\frac{t_{maxWT}}{r} \cdot \tilde{d} + t_{maxWT} \qquad (1)$$

$$\tilde{d} = \min\{d, r\}$$

where d : distance to sender

t_{maxWT} : maximum waiting time

r : transmission range.

To understand our motivation for waiting rather than resending the message immediately, consider the broadcast nature of radio waves. Multiple hosts can receive the same packet simultaneously. Then, an immediate resending would cause burst-like traffic on the channel. We try to avoid peak load by forcing the receivers to wait. Using the function WT, hosts at the border of the reception area are most likely to take part in forwarding the message quickly.

While the process awaits the moment to resend, it still updates the sets of neighbors and known senders. Then, it sub-

tracts the set of known senders from the set of neighbors. If on any of these updates this set difference becomes empty, the process switches into the "WaitForNeighbor" state. Otherwise, it forwards the message after the timer expires and the calculated waiting time is over.

If at the creation time of the process the set difference of neighbors without known senders is already empty, then there are no new receivers nearby and the process immediately enters the "WaitForNeighbor" state. In this mode, the process waits until an update of the set of neighbors occurs such that the set difference of neighbors without known senders is not empty anymore. Then, the system forwards the message. After forwarding the message, the process instance terminates.

The top layer *bTJam* implements the application and is in our example responsible for detecting a traffic jam. It issues the join and leave requests when the velocity of the vehicle exceeds or falls below certain values. Using the view on the local group membership, the block *bTJam* also classifies the vehicle as at the beginning or at the end of the congestion. If the group of slow vehicles has at least two members and the vehicle is the outermost group member, *bTJam* initiates a message to be sent to the other end of the traffic jam. In the case that the jammed vehicle is alone, it initiates sporadic messages about its position. Due to our routing scheme working in sparsely populated and fragmented networks, such a message might still reach another jammed vehicle in the same driving direction. If a vehicle at the border of the traffic jam receives a message from another borderline vehicle, it calculates the distance between the originator of the message and itself to detect the size and position of the traffic jam.

MICROSCOPIC TRAFFIC SIMULATION

To analyze the proposed protocols, we simulate realistic highway traffic that is prone to jam formation. We distinguish macroscopic and microscopic traffic models. Macroscopic models consider the dynamics of vehicle density and average velocity of road segments. However, in order to study the distributed system as a collection of protocol instances, our traffic simulation requires a microscopic model which determines the dynamics of individual vehicles. The challenge is to find a simple model of a single vehicle that leads to a jam formation from a macroscopic perspective.

In this article, we utilize the microscopic traffic model described by Krauß [14], which has been developed to study jam formation with a minimal driver model. The model for one vehicle consists of four parameters and four rules for each time step. These parameters are the maximum velocity v_{max}, the maximum acceleration a, the maximum deceleration b and the amount of noise ε that introduces stochastic behavior to the model. The time is discrete and ticks with an interval of $\Delta t = 1$ second. The spatial values of positions are continuous and one-dimensional.

The rules of the model describe how the vehicle chooses a new velocity v and applies it to reach a new position x in the next time step. First, the maximum velocity v_{safe} is computed so that the vehicle does not collide with the vehicle in front. Then, the desired velocity $v_{desired}$ is the minimum of either the maximum velocity in the model, or the current velocity plus the greatest acceleration possible in this time step, or the safe velocity v_{safe} to prevent the vehicle from driving into its predecessor. In the next rule, the actual velocity for the next time step equals the desired velocity minus a random term that reflects the inaccuracy human perception of speed and distance, if not negative. The formulae for the update rules are:

$$v_{safe} = v_l + \frac{gap - v_l \cdot \tau}{\tau_b + \tau} \quad \text{with } \tau_b = \frac{v_l + v_f}{2 \cdot b} \tag{2}$$

$$v_{desired} = \min[v_{max}, v_f + a \cdot \Delta t, v_{safe}] \tag{3}$$

$$v(t + \Delta t) = \max[0, v_{desired} - \varepsilon \cdot a \cdot \Delta t \cdot \mathrm{random}()] \tag{4}$$

$$x(t + \Delta t) = x(t) + v(t + \Delta t) \cdot \Delta t, \tag{5}$$

where v_l is the velocity of the leading vehicle, v_f is the velocity of the follower and gap is the gap between the leader and the follower.

We apply the microscopic traffic model to a highway scenario on a straight road. We had to set the number of lanes per driving direction to one because the extension of the traffic model towards multilane traffic is not trivial. For more than one lane, the model requires lane changes to cope with traffic in one lane influencing the traffic in a neighbor lane. Also, we do not address other phenomena like traffic congestions caused by curiosity of drivers in the opposite direction.

For traffic simulations, the traffic density ρ describes the number of vehicles per kilometer and lane. We want to cause the traffic jam in direction one (that is where the x-positions of the vehicles increase over time.) Therefore, we set ρ to a high value in this direction and to a medium value for the oncoming driving direction.

Initially, the traffic scenario spreads the vehicles' positions on the lane according to ρ. For higher density ρ, the chance that a traffic jam evolves is higher, but we cannot control when, where or if the jam grows. Therefore, we disturb the initial setting by stopping the first $k = 5$ vehicles. For smaller $k < 5$, the traffic simulation did not cause a traffic jam in every run. The vehicles will not stop for long because the foremost vehicle quickly accelerates again having no predecessor in its lane. Nevertheless, the intrusion is enough to reliably cause a traffic jam for $k = 5$.

Table 1 summarizes the parameters of the road and the traffic model, which are derived from Krauß [14]. The simulation captures a duration of fifteen minutes. The traffic jam simulator outputs trace files with the position of vehicles during time. These trace files provide the mobility model used for the ad hoc network simulation.

Table 1. Parameters of traffic model

Description	Value
Length of simulated road	10 km
Number of lanes per direction	1
Traffic density ρ	15 vehs/km
(in direction one and two)	5 vehs/km
Number of stopped vehicles k	5
Space occupied by vehicle	7.5 m
Maximum velocity v_{max}	36 m/s $\approx$ 80 mph
Maximum acceleration a	1.5 m/s^2
Maximum deceleration b	4.5 m/s^2
Influence of noise ε	1.5
Reaction time of driver τ	1 s

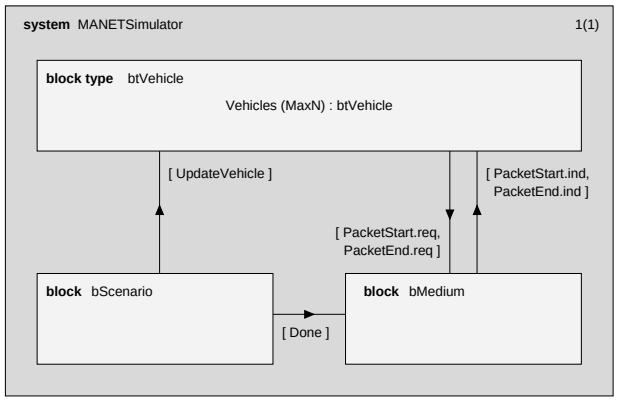


Figure 3. Overview of integrated simulator in SDL

AD HOC NETWORK SIMULATION

The ad hoc network simulation reproduces the behavior and the interaction of multiple communication devices. The simulator consists of the multiple modules for the vehicles equipped as described above and as seen in Figure 1, a module controlling the traffic scenario, and a channel module that organizes the exchange of messages. Figure 3 provides an overview of the integrated simulator.

The scenario module controls the simulation run. It processes the trace files generated by the traffic simulator and updates the positions of the vehicles periodically. When the end of the trace file is reached, the module shuts down the simulator properly.

The channel model assumes perfect reception of messages in transmission range and always drops packets outside the circular transmission area. Also, we model reception failures at stations that either send themselves or receive a message from another sender simultaneously. In the latter case when two or more transmissions overlap in time at the receiver, all of the messages are lost. Table 2 summarizes the main parameters of the ad hoc network simulation.

Table 2. Main parameters of network simulation

Description	Value
Transmission range r	600 m
Transmission duration of traffic jam message	5 ms
Transmission duration of group message	1 ms
Transmission duration of heartbeat	0.5 ms
Maximum number of hops	20
Maximum waiting time t_{maxWT}	40 ms
Heartbeat rate	1 s
Velocity to escape jam	70 km/h
Velocity to become jammed	40 km/h

METRICS AND RESULTS

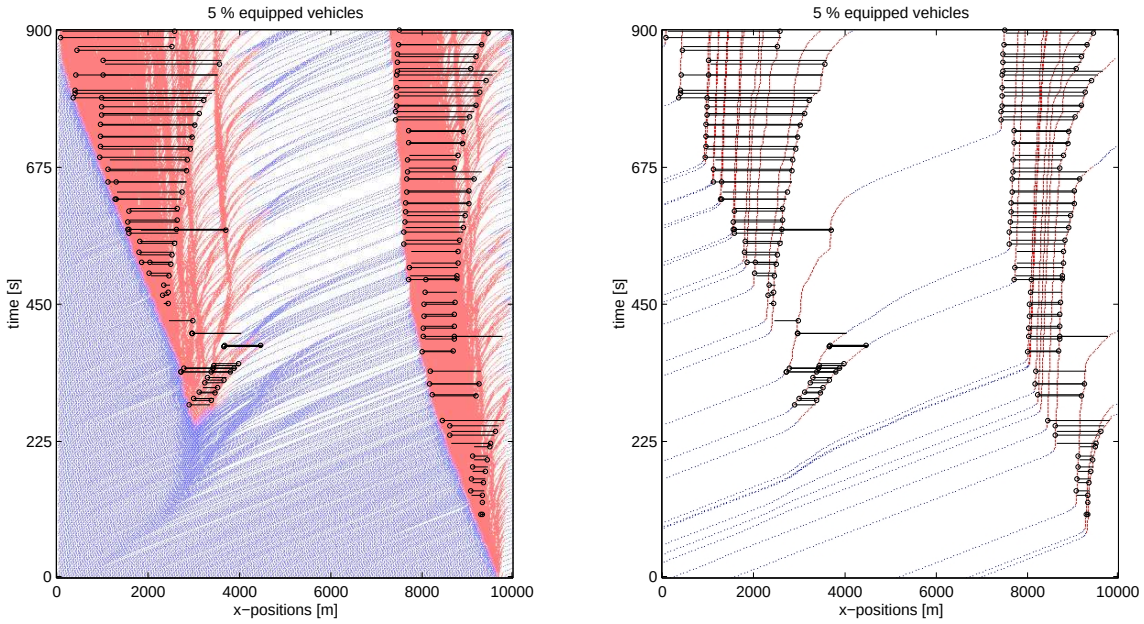
To evaluate the general behavior of the communication system, we compare the size and the position of the traffic jam as calculated by the vehicle with the actual size and position of the congestion on the road. For other metrics to study the layers below the application refer again to [13].

We distinguish between the situation with all vehicles on the road and the situation reduced to only the equipped vehicles. Figure 4 shows an example of these two situations. As the background, we use the space-time plot with all vehicles in Figure 4(a) and the space-time plot reduced to only the equipped vehicles in Figure 4(b). In space-time plots, we print one pixel at the corresponding position for all mobile entities per time step. Then, every vehicle leaves a trace of its movement as a curve consisting of single pixels. Fast vehicles create slopes according to their velocity, whereas slow or stopped vehicles remain in their position over time describing an almost vertical line.

The traffic jam at a jammed vehicle is detected when a message regarding the traffic jam is received and the vehicle classifies itself as being at the border of the jam. Then, the vehicle determines the jam size and position as the pair of its own position and the distance to the originator of the message. Each time an equipped vehicle detects a traffic jam, we plot a circle at its current position and a horizontal line of the length corresponding to the detected size of the jam.

The example in Figure 4 shows that the traffic jam detection for an evolved jam works fairly well with only 5% equipped vehicles on the road. Reduced to the situation with only equipped vehicles left, the two jams in this simulation run are detected almost perfectly by the outermost vehicles.

For every detection of a traffic jam, we compare the detected size and position with both types of jams—with the real jam that contains all vehicles, and with the equipped jam that is reduced to the equipped vehicles. Figure 5 provides an example of our metrics to measure the error of traffic jam detection. At the displayed time, the second equipped and jammed vehicle detects the jam by receiving a message from behind. Then, the absolute error of the real traffic jam size is the sum of lengths (1) and (2). The absolute error of the real



(a) all vehicles

(b) equipped vehicles

Figure 4. Example of traffic jam detection

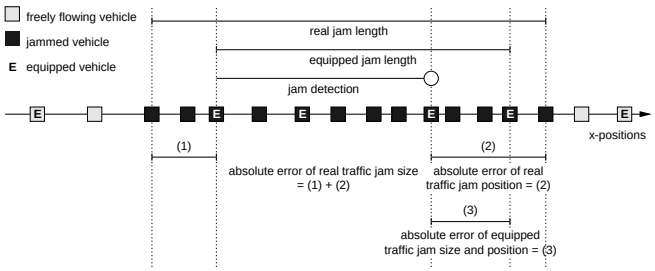


Figure 5. Example of error of traffic jam detection

Table 3. Results for error of traffic jam detection with confidence intervals for $\alpha = 5\%$

	Percentage of equipped vehicles [%]	Mean error of traffic jam size [m]	Mean error of traffic jam position [m]
all vehicles	1	2543 ± 253	1314 ± 123
	2	2475 ± 201	1287 ± 97
	5	2101 ± 180	1107 ± 87
equipped vehicles	1	742 ± 189	354 ± 89
	2	1102 ± 171	580 ± 80
	5	1327 ± 174	727 ± 83

traffic jam position is the distance (2) to the beginning of the jam. The corresponding measures for the equipped jam are the distance (3) to the actual first equipped and jammed vehicle. The end of the equipped traffic jam is detected perfectly.

As a free parameter of simulation runs, we varied the percentage of equipped vehicles on the road—three different settings were tested for 1, 2, and 5% of equipped vehicles on the road. For each parameter, we carried out 100 simulation runs capturing a simulated duration of 900 seconds.

Table 3 summarizes the results of the error of traffic jam detection for jams of all vehicles and for jams that contain only equipped vehicles, respectively, along with the confidence intervals. In the second and third column, we find the mean values of absolute errors of traffic jam size and position as a function of the percentage of equipped vehicles on the road.

For real traffic jams considering all vehicles on the road, both columns show the same characteristics. The functions reach their maximum for the smallest deployment of 1% equipped vehicles on the road—an average error of 2543 m for the traffic jam size and an average error of 1314 m for the traffic jam position.

The last three rows contain the results for traffic jam detection reduced to jams of equipped vehicles. In general, we achieve better results for jams that consist solely of equipped vehicles than for jams of all vehicles. The maximum of the absolute errors in traffic jam detection is reached for 5% equipped vehicles on the road and averages an error of 1327 m in size and 727 m in position. In contrast to the error metrics for all vehicles on the road, the average error of

detected traffic jams consisting of only the equipped vehicles increases slightly for very small deployments of 1% to 5% equipped vehicles on the road. We explain this fact with the example shown in Figure 6.

Diagram 6(a) displays a simulation run with 1% equipped vehicles on the road. Although only the three vehicles on the left hand side detect the traffic jam, they capture the size and the position perfectly. The single equipped vehicle on the right hand side lacks communication partners and hence never exchanges messages to detect the traffic jam.

With 2% equipped vehicles on the road—as seen in Figure 6(b), the equipped vehicles occasionally detect the traffic jam inaccurately. Equipped vehicles on the left hand side that receive a message from the jam on the right hand side via intermediate vehicles traveling in the opposite direction (not plotted) occasionally cannot distinguish the two jams and extend the error of detection in the overall mean value of our metric.

Finally, Figure 6(c) shows frequent, and mostly accurate, detection of the traffic jams. Still, having a higher frequency of detection also increases the average error of the detected jam. We circled the regions of frequent misclassification where the detected jam is either too long or too short.

CONCLUSION

We describe an ad hoc network protocol stack for an intervehicle communication system and apply it to detecting traffic jams on highways. We specify our algorithms using the Specification and Description Language (SDL) and a software tool for SDL. The software tool is able to simulate the behavior of SDL models. Therefore, we use the SDL tool also as an ad hoc network simulator. A microscopic traffic model determines the mobility of single vehicles and yields the scenario in which the ad hoc network is simulated.

The results of the ad hoc network simulation to detect traffic jams focus on sparse deployment of the intervehicle communication system. Such a situation is likely to occur shortly after the system is introduced to the market. Even with only 1% of the vehicles on the road being equipped, the detection of traffic jams is possible. Space-time plots of mobile hosts combined with drawings of routed messages visualize the powerful effect and robustness of rather simple, distributed protocols for communicating in ad hoc networks.

The approach herein to integrate simple protocols together with an application enables the evaluation of the complete communication system. Our algorithms for the single layers are much simpler than most of the recently proposed approaches, e.g. our medium access control resembles a carrier sense multiple access without back-off procedures if the channel is busy—we simply drop the packet. Still, we achieve visible results even when only a fraction of the network population is equipped with the system. The system model in the network simulator is easily adaptable to other applications by replacing the top layer in our communication system.

References

[1] Broch, J.; D.A. Maltz; D.B. Johnson; Y.-C. Hu; J. Jetcheva. 1998. A performance comparison of multihop wireless ad hoc network routing protocols. In *4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, Oct.

[2] Ko, Y.-B. and N.H. Vaidya. 2000. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, July.

[3] Karp, B. and H.T. Kung. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, Aug.

[4] Vahdat, A. and D. Becker. 2000. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-200006, Duke University, April.

[5] Hong, X.; M. Gerla; G. Pei; C.-C. Chiang. 1999. A group mobility model for ad hoc wireless networks. In *ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 53–60, Aug.

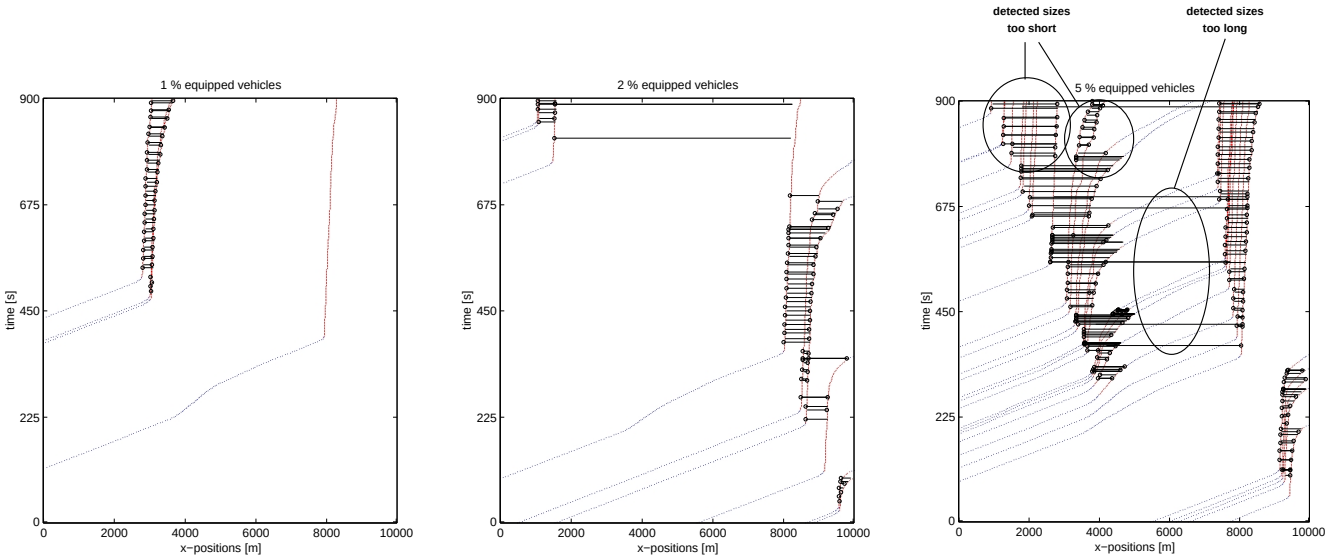
[6] Briesemeister, L. and G. Hommel. 2000. Overcoming fragmentation in mobile ad hoc networks. *Journal of Communications and Networks.*, 2(3):182–187, Sep. ISSN 1229-2370.

[7] Morris, R.; J. Jannotti; F. Kaashoek; J. Li; D.S.J. De Couto. 2000. CarNet: A scalable ad hoc wireless network system. In *Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System*, Kolding, Denmark, Sep.

[8] Briesemeister, L.; J. Donandt; L. Schäfers; A. Weidt. 1998. "Spread-Spectrum Funkmodems für den Einsatz zur direkten digitalen Fahrzeug-Fahrzeug Kommunikation." Technical Report FT3/AS-98-002, Daimler-Benz AG.

[9] Michael, L.B. et al. 1998. DS/SS inter-vehicle communication experiments in 2.4 GHz ISM band. In *IEEE International Conference on Intelligent Vehicles*, pages 397–401, Oct.

[10] Briesemeister, L.; L. Schäfers; G. Hommel. 2000. Disseminating messages among highly mobile hosts based on inter-vehicle communication. In *IEEE Intelligent Vehicles Symposium*, pages 522–527, Oct.



(a) rare detection with accurate traffic jam sizes

(b) detection with occasionally misclassified traffic jam sizes

(c) frequent detection with some inaccurate traffic jam sizes

Figure 6. Example of detection of traffic jams degrading with increasing deployment

[11] International Telecommunication Union. 1999. *ITU-T Recommendation Z.100. Specification and description language (SDL)*, Sep.

[12] Ellsberger, J.; D. Hogrefe; A. Sarma. 1997. *SDL – Formal Object-oriented Language for Communicating Systems*. Prentice Hall.

[13] Briesemeister, L. 2001. "Group Membership and Communication in Highly Mobile Ad Hoc Networks." PhD dissertation, Technical University of Berlin, Germany, Nov. http://edocs.tu-berlin.de/diss/.

[14] Krauß, S. 1998. "Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics." PhD dissertation, University of Cologne, April.

Linda Briesemeister received her graduate degree in Computer Science from the Technical University of Berlin, Germany in March 1998. Since then, she has worked on intervehicle communication at DaimlerChrysler Research and Technology as a research scientist. She obtained her Ph.D. degree in Engineering from the Technical University of Berlin in November 2001. Her research interests include distributed algorithms applied to mobile computing and wireless networks as found in intervehicle communication. She will be working for SRI International located in Silicon Valley from early 2002 on.

Günter Hommel received his degree Dipl.-Ing. in Electrical Engineering from T.U. Berlin in 1970. He first joined the Department of Electrical Engineering and then the Department of Computer Science at T.U. Berlin where he received his Ph.D. in Computer Science. In 1978 he joined the Nuclear Research Center in Karlsruhe working in the field of real-time systems. From 1980 he was responsible for a research group in Software Engineering at the German National Research Center for Information Technology (GMD) in Bonn. In 1982 he was appointed professor at the Institute for Computer Science of T.U. Munich, where he worked in the fields of real-time programming and robotics. Since 1984 he is professor at the Institute for Technical Computer Science of the T.U. Berlin. He now is the head of the Real-Time Systems and Robotics Group.