A Novel Approach to Evaluating Implementations of Location-Based Software

Daniel Herrscher, Steffen Maier, Jing Tian*, Kurt Rothermel
University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
Universitätsstr. 38, D-70569 Stuttgart, Germany
herrscher@informatik.uni-stuttgart.de

Keywords: performance analysis, network emulation, location-based software, positioning devices, mobile ad hoc networks.

Abstract

There are two main approaches to measuring the performance of location-based software for mobile ad hoc networks: live testing and measurements in an emulation testbed. The former is problematic because of high resource costs and low reproducibility. The latter offers the advantage of reproducible network conditions in the laboratory. In addition, location-based software has to be provided with appropriate location information. We develop virtual devices that mimic the interface and functionality of actual positioning devices, and integrate these into an existing network emulation testbed. The combination of mobile network emulation and positioning device emulation facilitates the evaluation of location-based software in different scenarios. We provide sample measurements using an existing location-based ad-hoc routing protocol implementation for inter-vehicle communication in a scenario with 50 emulated cars. The results are in good agreement with the available simulations.

I. INTRODUCTION

During the development of software for mobile devices, it is essential to analyze the impact of different scenario parameters on performance. These parameters usually include movement characteristics and network properties, such as communication range and bandwidth. In early design stages, performance is compared by mathematical analysis and simulation. Measurements are used to check the theoretical results as soon as implementations become available.

Comparative performance measurements in real environments are considered problematic for two reasons. First, in scenarios with mobile nodes and wireless networking, it is hard to obtain multiple comparable measurement runs. The node movements have a strong impact on the quality of wireless communication, and having several nodes exactly follow a predefined choreography is not straightforward. Secondly, resource requirements prohibit measurements in larger scenarios.

Network emulation testbeds allow these measurements to be conducted in the laboratory. These testbeds consist of a number of PC nodes connected by an emulated network. Special testbeds for mobile ad-hoc network (MANET) emulation control the network properties according to virtual node positions. The reproduction of the desired network properties is performed by *network emulation tools* running on each testbed node.

However, current testbeds are not suitable to evaluate *location-based software*. Software that needs location information has to access a positioning device, which is not physically present in a testbed. In this paper, we propose to integrate a *virtual positioning device* to an existing MANET emulation testbed. The virtual device exactly mimics the interface and functionality of a real positioning device. Software running on testbed nodes can access the virtual device through a serial interface, just like a real device. The position information that is provided by the emulated device matches the virtual position of the testbed node according to the emulation scenario. This facilitates the evaluation of location-based software in emulated environments in the laboratory.

To show the applicability of our approach, we present measurements with an existing location-based routing protocol implementation designed for inter-vehicle communication [1]. For a scenario with 50 cars driving around in a typical city center, we measure the overall delivery ratio for different transmission ranges. The results are in good agreement with the available simulations.

The remainder of this paper is organized as follows: Related work is presented in Section II. In Section III, we briefly present our approach to MANET emulation. Section IV describes the integration of the emulated positioning devices in our emulation infrastructure. Section V provides sample measurements. Section VI summarizes the paper.

* Jing Tian is funded by the European Project CarTalk2000, contract number IST-2000-28185

II. RELATED WORK

The evaluation of location-based software in real MANET environments suffers from two problems: Limited repeatability and high resource costs. To alleviate the repeatability problem, [2] proposes a MANET testbed consisting of several notebooks equipped with wireless network interfaces. According to a predefined choreography, the notebooks display movement commands to be followed by students carrying the notebooks. Although the original paper does not mention it, the approach would also be suitable for the analysis of location-based software if the notebooks were equipped with positioning devices. While the approach does provide repeatable node movements with limited accuracy, the repeatability of the resulting network properties is still not guaranteed due to other influences that cannot be eliminated. Of course, the resource cost problem remains.

Therefore, it is reasonable to build MANET testbeds that do not require to distribute the nodes physically. The basic challenge is to mimic the influence of virtual node mobility on the actual network properties. Hardware-based approaches use wireless networking devices with additional hardware to control signal attenuation, and thus to emulate the changing communication quality of the respective nodes [3, 4]. Because of the special resource requirements of these approaches, the software-based emulation of MANET properties is much more common.

Software-based MANET emulation approaches can be classified into two sub-categories: Centralized and distributed. The centralized approaches require all traffic in a testbed to be sent to a central instance. This instance maintains a node mobility model and decides which transmissions can be successfully delivered, based on the position and the transmission range of the emulated MANET nodes. The existing solutions for centralized MANET emulation [5, 6, 7] differ in the complexity of the network and signal propagation model that is considered by the central instance, but have in common that this instance constitutes a performance bottleneck limiting the overall network traffic in a scenario.

In distributed MANET emulation, there is still a central instance that maintains a mobility model, but the traffic processing is done by local emulation tools directly on the respective testbed nodes. This allows for much larger scenarios without traffic limitations. Both the central model and the local tools have impact on the realism of the emulation. Some approaches simply use the built-in firewall of the operating system as emulation tool [8, 9]. Because firewalls can be configured to drop frames depending on the sender address, this is sufficient to model the changing connectivity in a MANET. The introduction of dedicated emulation tools that run inside the communication stack facilitates to control more parameters, such as loss probabilities and delay [10]. Even the impact of collisions on a shared media network can be considered by some tools [11].

While one of the approaches to MANET emulation makes the virtual position information available on the nodes [9], this information is not accessible for user programs. The position information is only intended to be used by processes belonging to the emulation infrastructure, e.g. location-based load generators. To our knowledge, there is no existing positioning device emulation in network emulation testbeds.

III. MANET EMULATION

In this section, we briefly describe our approach to MANET emulation. We use a testbed running a software-based distributed emulation approach with a versatile emulation tool. First, we give an overview of the testbed architecture. Then, we explain the basic functionality of our MANET emulation tool.

A. Architecture

The testbed we use for the emulation of MANET environments consists of 64 Linux PC nodes connected by Gigabit Ethernet. For detailed information about the testbed architecture, please refer to [12]. Each PC node represents one mobile node in a MANET scenario and serves as execution environment for software to be evaluated. A custom emulation tool runs on each node. The tool inserts an additional *emulation layer* into the communication stack. This layer mimics the behavior of a wireless network, including the properties of both the wireless networking device and the underlying wireless channel.

A central *scenario controller* that runs on a dedicated high-performance PC is responsible for setting up and maintaining the scenario. It simulates the node mobility in real time according to a movement trace. Typical node movement traces for MANET scenarios can be generated on the basis of mobility models [13]. Since most available mobility model implementations generate movement traces in the file format used by the simulator ns-2 [14], we use the same format as input for our emulation scenarios.

The network properties for each node are calculated based on the nodes' virtual positions. Currently, we use a simplified model to derive the network properties: If two nodes are within a fixed transmission range, they can communicate via an emulated channel with parameters that are specified beforehand. If their distance exceeds the threshold, the connection is cut off completely. This model assumes uniform free-space radio propagation, which is the basis for most other approaches modelling MANET communication characteristics [15, 8]. We are currently investigating the integration of more sophisticated wave propagation models that consider spatial information, such as buildings or streets [16].

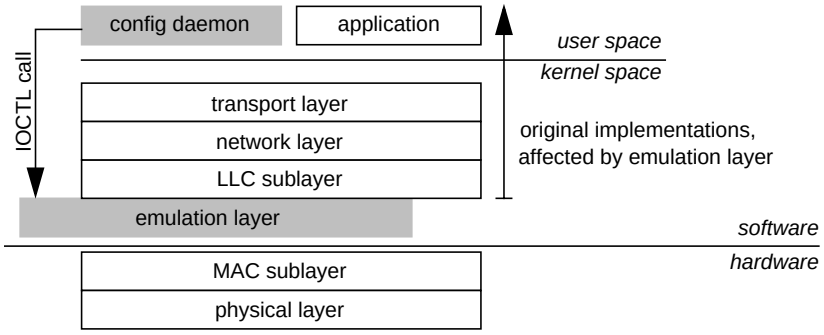

Figure 1. Location of the emulation layer in the protocol stack.

Because the scenario controller does not process the actual traffic in the emulation scenario, we cannot consider the effects of traffic on the network properties here, namely interference and collisions. These effects have to be addressed by the distributed emulation tools. In other words, the central instance computes which nodes have the *chance* to communicate based on their position. Depending on the actual traffic in a specific area, the emulation tools have to determine if an *actual transmission* can be received successfully.

The central scenario controller sends the current network parameters to a configuration daemon running on each node. We use simple UDP messages to send the parameter updates. An additional separated administration network connects the control PC to all testbed nodes to ensure that the frequent update messages do not interfere with traffic inside the emulation scenario. Upon reception of a parameter update, the daemon configures the actual emulation tool by I/O control calls (IOCTL) to the kernel.

The central scenario control eases the setup and management of scenarios and synchronizes the emulation tools, while the actual traffic processing is done in a distributed fashion. Thus, our architecture is scalable in terms of network traffic and number of nodes.

B. Emulation Layer

In order to emulate the properties of the physical layer and the medium access sublayer (MAC) of the data link layer, our emulation tool offers the same service abstraction as the MAC layer, i.e. an unreliable datagram service (see Fig. 1).

Starting with logical link control (LLC), the layers above the emulation layer experience the emulated network properties. For these layers, we can choose to use the original protocol implementations of Linux, or replace them with experimental implementations. The presence of the emulation layer is completely transparent to the above layers.

The emulation tool processes both egress and ingress traffic. According to the desired network parameters, the tool limits the bandwidth of egress traffic and adds delay to the frames. Ingress traffic is subject to a frame loss ratio, which can be specified separately for each sender address. As a consequence, it is possible to emulate the limited propagation range of a transmission in a MANET. A broadcast transmission issued by node N is physically delivered to all other nodes in the emulation testbed. On nodes within N's propagation range, the drop rate for N is set to 0%, and therefore incoming transmissions from N are passed to the upper layers. On nodes that are too far away from N according to the scenario, the emulation tool is configured to drop 100% of N's transmissions. This way, it is straightforward to implement "neighbor tables" on each node. Furthermore, intermediate loss rates can also be used to model the changing signal quality, depending on node distance and other factors affecting the quality (obstacles, etc.).

The current version of our MANET emulation tool does not yet consider the effects of interference and collisions. Therefore, the behavior of the emulated MANET is only realistic for scenarios with low network load. However, we are currently developing an improved version that can mimic these effects, based on an emulation of the MAC protocol specified in the IEEE 802.11b standard. In prior work, we already proved that the software-based emulation of a MAC protocol is feasible [11].

IV. VIRTUAL POSITIONING DEVICE

The emulation testbed described in Section III facilitates the performance analysis of distributed applications and protocols designed to run in MANET environments. However, if the software under test is location-based, it needs to access a positioning device. We cannot assume that this resource is physically present in a testbed. Instead, we have to provide the software under test with a *virtual positioning device.* The device provides position data that is consistent with the virtual node position according to the emulation scenario.

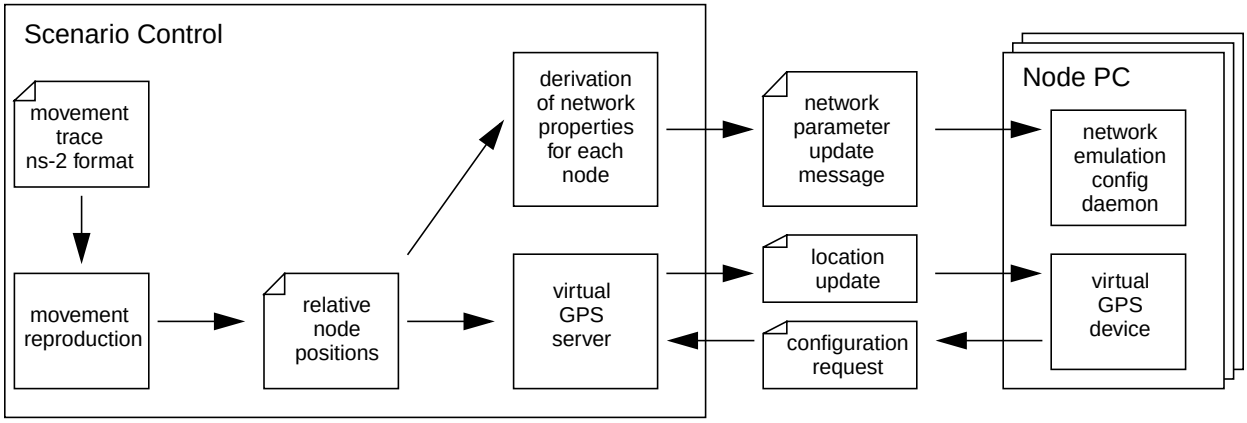

Figure 2. Network parameter and location updates.

A. Interface

There are many positioning and tracking technologies that a MANET node could use in reality to obtain its position. Because of its general availability, however, most existing systems for outdoor use rely on GPS as positioning system. The standard application interface for GPS devices has been defined by the National Marine Electronics Association (NMEA), and is referred to as the "NMEA 0183 standard" [17].

Most GPS devices can be connected to a serial port. Even devices designed as interface card for PCI or PCMCIA slots are accessed by (virtual) serial ports. For that reason, we provide a *virtual GPS device* implementing the NMEA 0183 standard that talks through a *virtual serial port*. This way, we exactly implement the interface of a real GPS device. Consequently, existing location-based applications can interact with our virtual GPS device just like with any other GPS device.

B. Implementation

As described in Section III, we maintain a model of the node positions to derive the network properties. Therefore, we can use the same position information as input for the virtual positioning devices. Because we use movement traces in ns-2 format to specify node movement, we work with the same relative node coordinates as ns-2. By defining an offset for this coordinate system, we map the relative coordinates to meaningful absolute positions in the global WGS84 format used by GPS.

Similar to the real-time updates of the emulated network parameters, the administrative network of our testbed is used to send the position information from the central model to the respective nodes via UDP packets (Fig. 2). On each node, an instance of the virtual device processes the position updates. To applications, this instance behaves exactly like an actual GPS device that is connected to a serial port. The virtual device outputs NMEA 0183 messages including the current position and other information (time, speed, etc.) at a regular time basis.

GPS devices do not only output information in NMEA 0183 messages, they can also process configuration messages that are sent to the device. These messages include subscription information for special NMEA 0183 messages, the setting of waypoints, etc. Our virtual device can handle a superset of NMEA 0183 messages that are considered by several real GPS devices.

Our primary implementation goal for the virtual positioning device was to act exactly like a real positioning device, i.e. output and process NMEA 0183 messages on a serial port. In UNIX like operating systems, such as Linux, serial ports are accessed through *character device special files* in the "/dev" directory. We created a special file looking like a serial port, and configured it to redirect all accesses to this pseudo file to our virtual device implementation (Fig. 3). By this technique, any software accessing the virtual device perceives exactly the same behavior as with a real device.

V. SAMPLE MEASUREMENTS

In order to test the applicability of our emulation environment for performance evaluation of location-based software, we conducted measurements with a location-based routing protocol that was developed as part of the CarTalk 2000 project [1]. The routing protocol especially addresses inter-vehicle communication. During the development process, first of all a simulation model for ns-2 has been implemented. After simulations with ns-2, the protocol was implemented on the Linux platform. Preliminary tests were conducted using five cars with laptops, IEEE 802.11b wireless LAN adapters, and GPS receivers. Finally, we used our emulation testbed to evaluate the performance of the implementation in a larger scenario with 50 emulated cars.

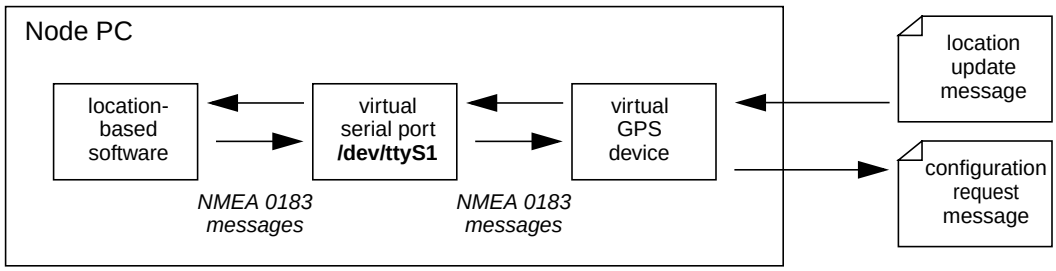

Figure 3. How to mimic a GPS device.

In this section, we briefly explain the routing algorithm, greedy location-based routing, and show some implementation issues. Then, we describe the scenario for our measurements. Finally, we present the measurement results we obtained from our testbed, and compare them to the performance predictions from the ns-2 simulation model.

A. Greedy Location-Based Routing

A number of special routing algorithms exist for MANETs. While it is possible to do MANET routing without using location information, the efficiency in highly mobile scenarios can be improved significantly by using the node location as additional information [18].

The nodes participating in location-based routing determine their own positions through positioning devices. Each node issues local broadcasts with its position ("beacons") periodically. Based on the beacons that a node receives, it can maintain an up-to-date neighbor table including the positions of the neighbors.

In contrast to traditional routing strategies that are based on symbolic addresses, the destination location forms the basis for the packet forwarding strategy. The sender has to use a location service to resolve symbolic addresses to locations.

Greedy forwarding with MFR (*Most Forward within Radius*) is a widely used forwarding strategy for location-based routing [19]: From the set of current direct neighbors, the protocol selects the node that is geographically closest to the destination as the next hop. Admittedly, greedy forwarding is not the ideal strategy for location-based inter-vehicle routing. The forwarding efficiency can be further improved by using additional spatial environment information [21].

B. Implementation

Greedy location-based routing with MFR has been implemented as part of the CarTALK 2000 project [20]. The routing protocol implementation runs on standard Linux in user space, and uses raw sockets to send and receive frames directly on data link layer. Each node obtains its current position from a GPS device. The routing core makes the packet forwarding decision based on the packet's destination position and the local neighbor table, which is constantly updated by a beaconing process. If a unicast packet queued for transmission by a local application does not have a destination location yet, the protocol queries a location service. The current location service implementation uses simple flooding to disseminate location request messages. The destination sends back location reply messages by location-based unicast to the original sender. To minimize the expensive flooding operations, each node maintains a location cache.

C. Sample Scenario

We set up a sample scenario with 50 cars, moving around in the city center of Stuttgart (2500 m × 1800 m) according to a graph-based mobility model [13]. Initially, the cars are placed randomly on the graph. During the scenario run time of 5 minutes, they move to a random target point on the graph, with a speed ranging from 5 to 20 m/s. After a random pause time of up to 20 s, they move to the next target point. Each car randomly selects a communication partner and sends constant bitrate traffic to it (one 64-byte packet per second). We measured the overall delivery ratio while we varied the transmission range between 50 m and 400 m.

D. Results

Fig. 4 shows the measurement results we obtained from our emulation testbed, compared to the performance predictions from the ns-2 simulation model. Each point in the figure represents the mean value of three simulation or emulation runs, respectively. Because the connectivity of the nodes and therefore the delivery ratio highly depends on the node movement, we generated three different movement patterns for the three runs, and used the same set of movement traces for emulation and simulation. The results reveal that the implementation shows the same qualitative behavior as predicted by the simulation model. However, the delivery ratio of the actual implementation stays below the values of the simulation. This can be explained by implementation details that were not part of the idealized simulation model. For example, the implementation uses a location service based on flooding, while the

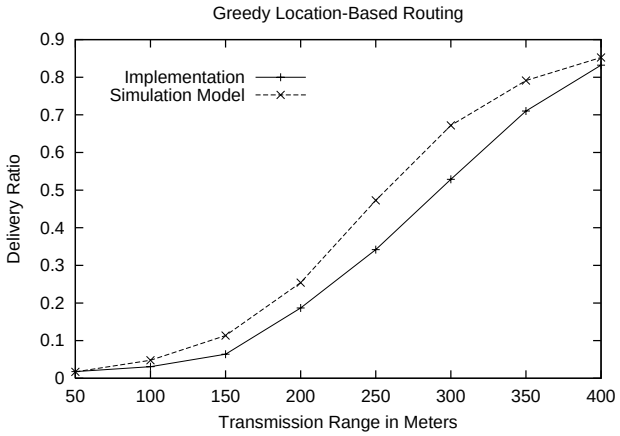

Figure 4. Performance in simulation model vs. performance in emulated environment.

simulation model queries an omniscient location service object for that reason.

Our conclusion from the sample measurements is that performance evaluations in emulated network environments produce reasonable results for *implementations* which are comparable to the results simulators produce on the basis of *simulation models.*

VI. SUMMARY

Performance evaluation of location-based software in real MANETs hardly provides reproducible results and makes high resource demands. Therefore, it is reasonable to run the evaluations in an emulated environment. While existing emulation testbeds can emulate the characteristic network properties of a MANET to some extent, location-based software needs to access a positioning device, which is not physically present in the testbed.

In this paper, we proposed an approach to mimic both the interface and the functionality of a positioning device. The emulated positioning device is seamlessly integrated into a MANET testbed, i.e. network emulation and positioning device emulation work on the same node movement model. The virtual device can be accessed through a serial port exactly like a real device, and outputs and understands NMEA 0183 messages, which is the standard format for GPS devices. Thus, to an application, it is completely transparent that the device is not real.

To show the applicability of our approach, we measured the performance of an existing implementation for location-based inter-vehicle routing, and compared the results to the predictions of a simulation model. The promising experiences from the sample measurements in our testbed lead us to the conclusion that our approach to performance evaluation of location-based software is a valuable complement to the traditional approaches simulation and live measurement.

REFERENCES

[1] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "CarTALK 2000 – Safe and Comfortable Driving Based Upon Inter-Vehicle-Communication," in *Proceedings of the IEEE Intelligent Vehicle Symposium*, Versailles, France, June 2002.

[2] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, "A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2002)*, pp. 412–418, Orlando, 2002.

[3] J. Kaba and D. Raichle, "Testbed on a Desktop: Strategies and Techniques to Support Multi-hop MANET Routing Protocol Development," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01)*, pp. 164–172, Long Beach, CA, Oct. 2001.

[4] E. Hernandez and A. Helal, "RAMON: Rapid-Mobility Network Emulator," in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pp. 809–817, Tampa, Florida, Nov. 2002.

[5] Q. Ke, D. Maltz, and D. Johnson, "Emulation of Multi-Hop Wireless Ad Hoc Networks," in *Proceedings of the 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000)*, Tokyo, Japan, 2000.

[6] J. Flynn, H. Tewari, and D. O'Mahony, "A Real-Time Emulation System for Ad Hoc Networks," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, pp. 115–120, San Antonio, Texas, Jan. 2002.

[7] T. Lin, S. Midkiff and J. Park, "A Dynamic Topology Switch for the Emulation of Wireless Ad Hoc Networks Using a Wired Network," in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02) (Wireless Local Network Workshop)*, pp. 791–798, Tampa, Florida, Nov. 2002.

[8] Y. Zhang and W. Li, "An Integrated Environment for Testing Mobile Ad-Hoc Networks," in *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02)*, pp. 104–111, Lausanne, Switzerland, June 2002.

[9] W. Chao, J. Macker, and J. Weston, "NRL Mobile Network Emulator," Naval Research Lab Formal Report 5523--03-10,054, Feb. 2003.

[10] W. Liu and H. Song, "Research and Implementation of Mobile Ad Hoc Network Emulation System," in *Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC'02)*, Vienna, Austria, 2002.

[11] D. Herrscher, S. Maier, and K. Rothermel, "Distributed Emulation of Shared Media Networks," in *Proceedings of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2003)*, pp. 226–233, Montréal, Canada, July 2003.

[12] D. Herrscher and K. Rothermel, "A Dynamic Network Sce-

nario Emulation Tool," in *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN'02)*, pp. 262–267, Miami, Oct. 2002.

[13] J. Tian, J. Hähner, C. Becker, I. Stepanov, and K. Rothermel, "Graph-based Mobility Model for Mobile Ad Hoc Network Simulation," in *Proceedings of the 35th Annual Simulation Symposium (ANSS-35 2002)*, pp. 337–344, San Diego, California, Apr. 2002.

[14] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation," in *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

[15] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pp. 85–97, Dallas, Texas, Oct. 1998.

[16] F. M. Landstorfer, "Wave propagation models for the planning of mobile communication networks," in *Proceedings of the 29th European Microwave Conference*, vol. 1, pp. 1–6, Munich, Germany, Oct. 1999.

[17] National Marine Electronics Association, "The NMEA 0183 Interface Standard, Version 3.01," available from NMEA, 7 Riggs Ave., Severna Park, MD 21146, USA, http://www.nmea.org, Jan. 2002

[18] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," in *IEEE Network Magazine*, vol. 15, no. 6, pp. 30–39, Nov. 2001.

[19] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," in *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, Mar. 1984.

[20] J. Tian, C. Maihoefer, M. Nelisse, M. Provera, I. Dagli, M. Tepfenhart, and C. Brenzel, "CarTalk2000 Deliverable D7: Routing Protocol Implementation," available from http://www.cartalk2000.net, Oct. 2003

[21] J. Tian, L. Han, K. Rothermel, and C. Cseh, "Spatially Aware Packet Routing for Mobile Ad Hoc Inter-Vehicle Radio Networks," in *Proceedings of the 6th IEEE International Conference on Intelligent Transportation Systems (ITSC 03)*, pp. 1546–1551, Shanghai, China, Oct. 2003.