Simulating Mobile Ad-Hoc Networks in City Scenarios

Illya Stepanov, Kurt Rothermel

Institute of Parallel and Distributed Systems (IPVS), Universität Stuttgart
Universitätsstr. 38
70569 Stuttgart, Germany
{stepanov, rothermel}@informatik.uni-stuttgart.de

***Abstract* — Simulation tools are frequently used for performance evaluations of mobile ad-hoc networks. Currently the tools poorly support urban scenarios, since they do not take a spatial environment into account. In this paper, we describe a platform for the modeling of city scenarios. We extend ns-2 with corresponding mobility and wireless transmission models. By using its emulation facility, we integrate unmodified applications and real implementations of network protocols. We demonstrate the usefulness of the platform for performance evaluations by modeling a mobile application in a simulated environment of Stuttgart downtown. We show that it helps identifying application problems before deployment.**

***Index Terms* — Communication systems, mobile communication, modeling, simulation**

I. Introduction

Mobile ad-hoc networks (MANETs) are formed by wireless peers without relying on a fixed infrastructure. The devices communicate directly with each other while they are in transmission range. A typical communication technology in MANETs is IEEE 802.11 [28].

Many usage scenarios have been proposed for MANETs in city areas, e.g., Usenet-on-the-Fly [2], CarTALK 2000 [6], and Ad- Hoc City [18]. Network simulation tools [4], [12], [23] are frequently used for performance analysis. They have three common shortages.

First, they typically offer only simple user mobility models. For example, the widely used random waypoint mobility model [5] simulates the straight movement between randomly chosen points of the area. The model does not consider spatial constraints of the area like roads. It also neglects user travel decisions and realistic movement dynamics.

Secondly, the tools rely on rather simple wireless transmission models. Such models (e.g., Friis free space model [8] or the two-ray ground model [11]) assume an obstacle-free area and a line-of-sight between all communicating partners. As a consequence, the communication range is modeled by a circle around a mobile device. It is assumed that other devices residing within this circle receive the transmitted frames without errors. Communication with the devices beyond the circle is not possible. This model poorly reflects radio wave propagation in typical outdoor environments such as cities, in which buildings significantly affect the communication between mobile devices. The usage of more realistic radio propagation models changes simulation results considerably [24].

Thirdly, the network simulation tools use own (simplified) implementations of the network protocol stack. They differ from modules of real operating systems. Simulators do not execute real applications either. The applications need to be reprogrammed in order to fit a simulator's API. As we show in this paper, this hides many factors that influence the performance of applications in realistic situations.

In this paper, we extend ns-2 [4] for the modeling of city scenarios. We choose ns-2, since it is a major if not the most frequently used MANET simulator in our community. We integrate more realistic mobility and wireless transmission models. They consider a road network and radio propagation obstacles taken from a digital map of the area. We also use a fine-grained model of transmission errors, which is based on measurements of an IEEE 802.11 card manufacturer [17]. Ns-2 also provides an emulation facility for injecting traffic from real networks. We use it for integrating unmodified mobile applications and real implementations of network protocols running in separate virtual machines. We demonstrate the usefulness of the platform for performance evaluations by modeling a MANET application in a simulated environment of Stuttgart downtown. We show that it helps identifying application problems before deployment. Our implementations are publicly available[*].

The remainder of this paper is structured as follows. In Section II, we briefly describe our approach to model user mobility in city areas. Section III describes a more realistic wireless transmission model. Section IV demonstrates the integration of real applications and protocols into ns-2. In Section V, we describe a MANET application, which we use for our evaluations. Section VI describes our simulation scenario. We analyze simulation results in Section VII. Section VIII gives an overview of related work. Finally, Section IX concludes the paper.

II. Modeling User Mobility

Our approach to model mobility of users in city areas (Figure 1) is described in [25]. It reflects the following key

[*] http://www.ipvs.uni-stuttgart.de?id=illya.stepanov&lang=en

factors that impact user movements:

- City environment with points of interest and movement constraints (spatial model)
- User travel decisions (user trip model)
- User movement dynamics (movement dynamics model)

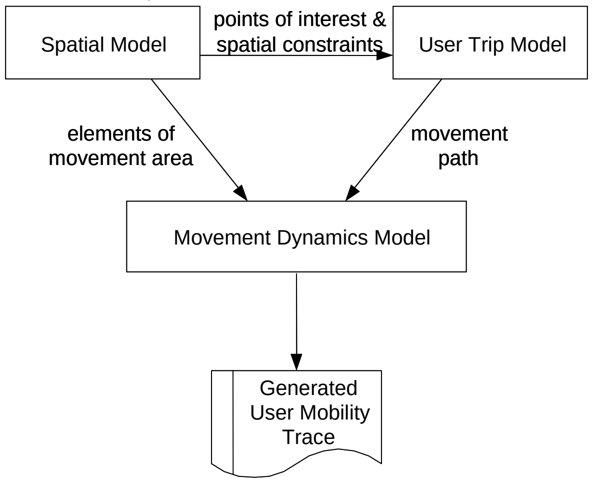

Spatial Model
points of interest & spatial constraints
User Trip Model
elements of movement area
movement path
Movement Dynamics Model
Generated User Mobility Trace

Figure 1: Design of the mobility model

The spatial model contains elements of a city environment. Some of them, such as streets and roads, constrain movements of users. Another group consists of the so-called “points of interest” (e.g., supermarkets or museums) that serve as destination points of movement. For each element, its properties (e.g., shop opening time, road speed limitation) and geometry are stored. The latter is used for constructing a street network graph. The spatial model can be initialized from digital maps in various formats, e.g., GDF [9] or GML [10].

Obviously, people do not move completely random in the target area. According to the activity-based travel demand approach [21], people move to perform an action in certain places, for example, shopping in particular shops or visiting predefined sights. A sequence of such actions (trip sequence) describes user movements in the area. The user trip model contains all the trip sequences that users perform during the simulation. It also performs movement path selection, e.g., using approaches from discrete choice theory [3].

In addition, mobile clients exhibit different movement dynamics. For example, pedestrians tend to move at lower speeds with frequent interruptions, while vehicles move at higher speeds and influence dynamics of neighboring vehicles. The movement dynamics model uses approaches from transport planning, physics, and vehicular dynamics [13], [27] to obtain user position changes along his/her movement paths. The position changes constitute a mobility trace, which is used as an input for MANET simulation tools, such as ns-2.

III. Modeling Wireless Transmission

The modeling of wireless transmission in MANETs includes the following steps: determination of signal receive power, computation of noise and interference, and packet reception [26].

Each time a mobile node transmits a frame, a simulator uses a radio propagation model to compute the signal receive power for every potential receiver. The result depends on attenuation that the signal experiences during propagation, e.g., due to environment. The noise and interference is the sum of powers of other signals and the receiver thermal noise. The signal to interference and noise ratio (SNIR) is the ratio of the signal receive power computed to the noise and interference. It has a correlation with bit-error rate of the received frame. The latter is used together with the frame length to estimate a probability of successful frame reception.

To model radio propagation in city areas, we rely on “intelligent ray tracing” model [29]. It considers a geographic map of the simulation area. In order to accelerate the performance of ray tracing, the model preprocesses the digital map and computes visibility relations between walls. Thereby it is about 1000 times faster than the classical ray tracing approach. The accuracy of the model is proven by measurements in European cities. For Stuttgart downtown, the mean error is 0.3 dB and the standard deviation is 5.8 dB [15].

The full details of integrating the model into ns-2 are given in [24]. We use a commercial implementation of the intelligent ray tracing model (WinPROP by AWE Communications [1]). For any given sender position (and other constant parameters like sender height, transmission power, wavelength etc.), WinPROP computes a map of receive power values for a grid, representing possible positions of a receiver. In our simulations, we use a 5 m $\times$ 5 m grid, which is the smallest grid size we could handle (smaller grid sizes would require much longer computation time and more disk space). We performed a separate investigation to assure that the chosen grid size has minor impact on simulation results.

For optimal performance, we precalculated the receive power values for each possible sender-receiver pair and stored them in a database. Each time ns-2 needs a receive power value, our radio propagation module reads the appropriate value from the dataset. To reduce the data access overhead, our module uses a caching strategy. As a result, the overall ns-2 simulation time with our module is comparable to the simulation time with a simpler model, such as two-ray ground.

Ns-2 decides on successful packet reception by only checking if a frame’s receive power is above or below the receive threshold of the network equipment. In order to perform more realistic simulations, we also use a fine-grained model of wireless transmission errors.

The model is based on measurements of a card manufacturer [17]. They correlate frame’s bit-error rate with the signal-to-noise ratio and the modulation scheme at the given transmission speed. We use the implementation from [30]. It models errors upon transmissions of control and data frames. The implementation determines target bit-error rate from a table using the transmission speed and the computed signal-to-noise ratio as indexes.

IV. Integrating Real Mobile Applications

Next, we integrate MANET applications into ns-2 (Figure 2). The simulator has an emulation facility (nse) allowing real network traffic to pass through it. We start mobile applications on separate computers corresponding to individual network users. In order to simulate more users than the number of physical hosts that we have, we use User-mode Linux (UML) [7] virtual machines. They run as user processes on top of operating system of physical hosts. A virtual machine appears as a single computer with own network interface for an application being executed inside of it.

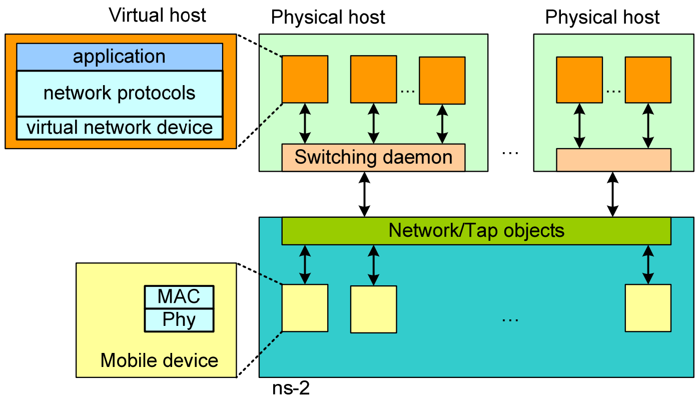


Figure 2: Integrating real mobile applications into ns-2

Layer 3 traffic of virtual machines is injected into ns-2. On physical hosts, the traffic is received by switching daemons, which is a standard way of providing the UML hosts with an access to outer network. We modify switching daemons to tunnel the traffic to “network” and “tap” objects of ns-2. The “tap” objects inject the traffic into simulated mobile devices. Ns-2 models the physical layer and the data link layer of Wireless LAN, as well as the mobility of network users. The packets received by the simulated devices are tunneled to the corresponding virtual hosts in a similar fashion.

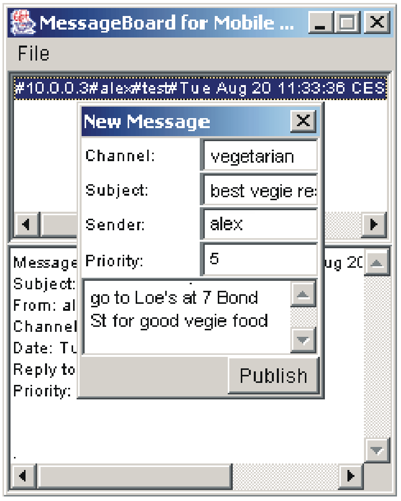
This described approach to traffic capturing allows the integration of unmodified MANET applications and real protocol implementations into ns-2. In future, it should be possible to replace UML with emulators of mobile devices (e.g., mobile phones, PDAs) to support other mobile platforms.

Virtual hosts and ns-2 perform in real time. We are confident that running several virtual hosts on one physical host and the centralized emulation of ns-2 do have impact on obtained simulation results. To assure that we stay below system load limit, we monitor packet drops at the ends of tunnel and CPU load of physical hosts. To improve the accuracy of ns-2 real-time scheduler, we use the extensions from [20]. They also provide those “network” and “tap” objects that serve as a basis for our implementation.

V. Sample MANET Application: Usenet-on-the-Fly

We use the described platform for simulating MANET applications in city scenarios. Here we describe relevant aspects of Usenet-on-the-Fly [2], which we use in this paper.

Usenet-on-the-Fly is an implementation of the well-known Usenet system for ad-hoc networks [16]. The corresponding client application (Figure 3) is implemented in Java. Hence, it runs on various hardware platforms, in particular, on PDAs. The graphical user interface (GUI) allows subscribing/unsubscribing to newsgroups (channels) and posting (publishing) of new messages. If a user posts a message to a specific channel, all subscribed users will receive it. Each message is distinguished by a unique ID.



#10.0.0.3#alex#test#Tue Aug 20 11:33:36 CES

Channel: vegetarian
Subject: best vegie re
Sender: alex
Priority: 5

go to Loe's at 7 Bond St for good vegie food

Message
Subject:
From: al
Channel
Date: Tu
Reply to
Priority:

Figure 3: Usenet-on-the-Fly screenshot

We assume that mobile users carry PDAs. The devices are equipped with an IEEE 802.11 Wireless LAN card. Every device is identified with a unique IP address. Unlike in wired networks, MANET devices must cope with limited connectivity and frequent topology changes. Hence, the messages are disseminated through diffusion. This involves periodical exchange between the devices that are in transmission range.

The application is multithreaded. Besides the GUI thread, the application uses 3 additional threads: “broadcast sender”, “broadcast receiver”, and “SOAP engine”.

The “broadcast sender” periodically broadcasts the device’s IP address. The mobile devices that are in transmission range receive the packet. Thereby the devices learn about other devices in their proximity. It is important to note that broadcast transmissions are unacknowledged as of the IEEE 802.11 standard. Hence, in the case of collisions there are no retransmissions performed.

The “broadcast receiver” receives the broadcasted IP addresses from neighbors. Upon receiving an address, the thread algorithm issues a Simple Object Access Protocol (SOAP) call “Get All Message IDs” addressed to the neighbor. The call returns the IDs of all the messages that are stored at the neighbor. Then they are compared with the receiver’s own messages. If the neighbor has any new messages, the “broadcast receiver” issues a SOAP call “Get Message Bodies”. The IDs of the required messages are passed as parameters. The received messages are added to the local database. In the current implementation, the database stores all the messages. The GUI performs the necessary filtering to display only the messages from the subscribed channels.

The “SOAP engine” thread processes sequentially incoming SOAP requests, such as “Get All Message IDs” and “Get Message Bodies”.

The application has the following parameters:

- *BroadcastInterval*: time interval between two successive broadcasts of a device’s IP address,
- *ServerSocketTimeout*: TCP timeout that the SOAP engine waits for a client to establish a connection,
- *ClientSocketTimeout*: TCP timeout that the SOAP engine waits for a request to be fully received,
- *SOAPTimeout*: TCP timeout that the “broadcast receiver” waits for a response to be received. This includes the time for connection establishment, message delays for transmitting request and response, and message processing on the remote side.

VI. Simulation Scenario

We simulate Usenet-on-the-Fly in Stuttgart downtown. We use a digital map 1.5 km × 1.5 km (Figure 4).

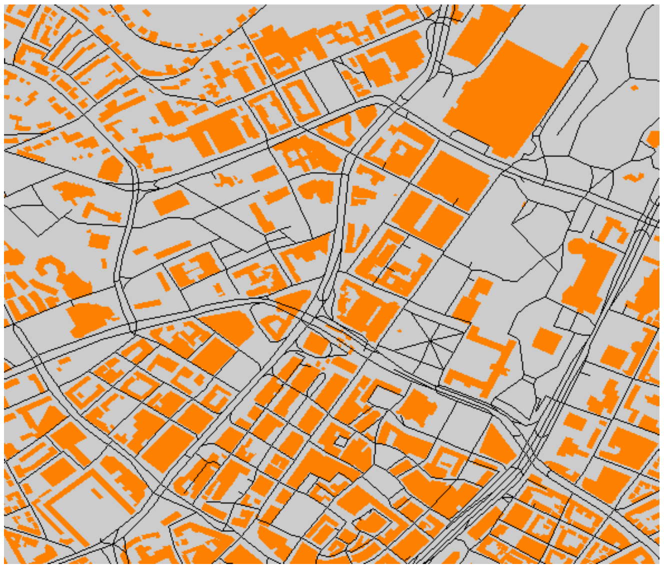


Figure 4: Map of the simulation area

In the simulated snapshot of city life, pedestrians move between different “points of interest”, like shops, restaurants, or museums. Since we currently do not have access to real human behavior data, we randomly generate user trips between these locations. The movement paths are obtained from the linked topological graph of the area using the Dijkstra shortest-path algorithm (we assume the users move along the shortest path to the destination). Upon arriving to a “point of interest”, the user stays there for duration between 10 and 15 minutes. Then the movement to the next trip point is initiated. The movement speeds are between 0.56 and 1.74 m/s [14]. They follow Gaussian distribution with mean 1.34 m/s and standard deviation 0.26 m/s [27]. The chosen speeds are kept constant during the movement between two points.

The simulated users carry mobile devices running Linux (the same operating system is started on our virtual hosts). The devices are equipped with wireless LAN cards. We use the parameters of Compaq© iPAQ 3660 PocketPCs and ORiNOCO© 802.11b WLAN cards in our simulations (Table 1). Since ns-2 does not support automatic switching between transmission speeds, we use the fixed speed of 1 Mbps.

The simulated message postings resemble 2-hour traffic from a real newsgroup (free-time activities). It consists of 18 postings. We assume that all the users are subscribed to this channel (newsgroup). The total simulation time is set to 8100 s, so the messages posted at the end have a chance to get spread in the network.

We create 5 virtual hosts on each physical host. The number was chosen in accordance with the number of mobile users and the number of physical hosts available. In spite of 5 parallel running virtual machines, CPU load of virtual hosts was low (nearly 0%), since the Usenet-on-the-Fly application performs little computational activity.

In simulations with more than 100 users, we noticed significant packet drops at the simulation front-end running ns-2. This makes communication between mobile hosts impossible even if they are in transmission range. The reasons for this are: 1) high network traffic load from virtual hosts, and 2) absence of flow control, since we used UDP for the tunnel between the switching daemons and ns-2. We will improve it in the future. Currently we support up to 100 of mobile users.

Table 1: Simulation parameters

Simulation time	8100 s
Number of mobile users	30, 50, 75, and 100
iPAQ battery capacity	3.7 V * 1350 mAh
Transmission power	15 dBm
Radio frequency	2.442 GHz
Transmission speed	1 Mbps
WLAN power consumption	idle mode: 0.045 W receive mode: 0.925 W transmit mode: 1.425 W
Broadcast interval	1, 2, 5, 10, and 20 minutes
Server socket timeout	10 s
Client socket timeout	20 s
SOAP timeout	30 s

VII. Simulation Results

In our simulations, we are mostly interested in analyzing message spreading with time. We define the spreading as a ratio of mobile users who received the posting. Figure 5 and Figure 6 present the results for different numbers of network users and broadcast intervals. The charts present the average for 18 postings.

The curves differ from common considerations, e.g., [2], which imply that a message spreads faster between more network users. Our results show the opposite. Moreover, the fact that a message spreads faster with less broadcasts seems to be irrational.

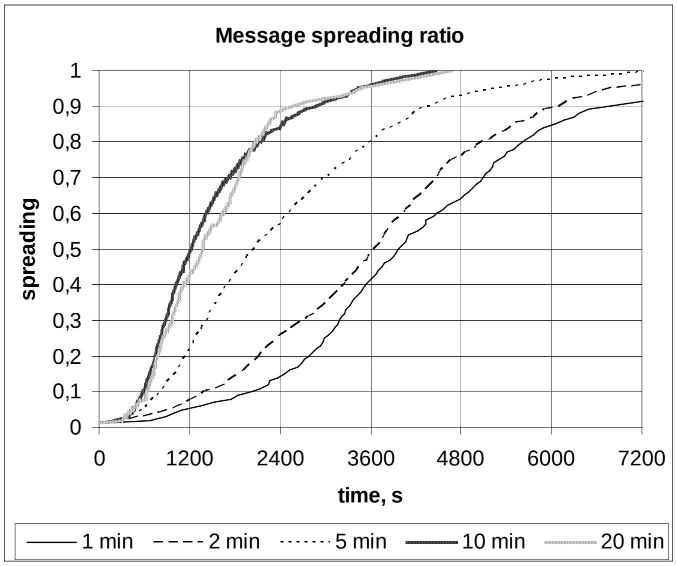


Figure 5: Message spreading ratio between 100 mobile users for different broadcast intervals

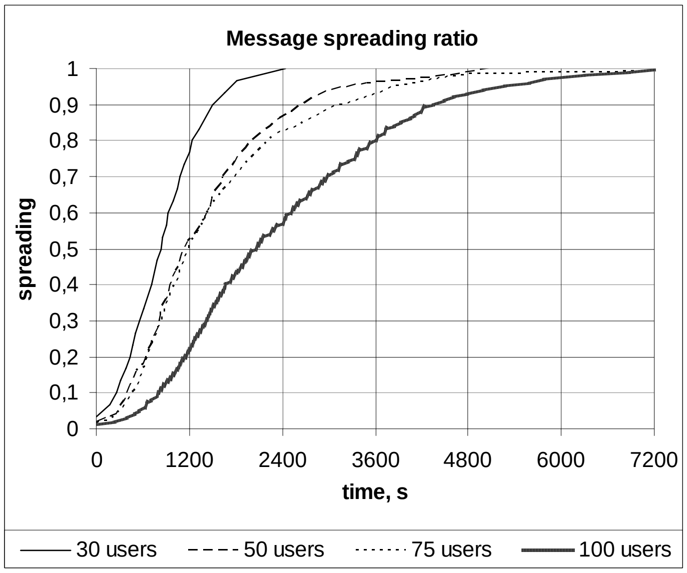


Figure 6: Message spreading ratio for different numbers of mobile users. Broadcast interval is 5 minutes

Precise investigation of application logs pointed out the problem. The operating system buffers the received broadcasts. The "broadcast receiver" processes them one after another. The "SOAP Timeout" parameter controls how long the thread waits for a SOAP response (performing necessary retransmissions, since SOAP relies on TCP). The default value is 30 seconds. However, according to Figure 7, in more than 60% of cases two users spend less than 30 s in each other's transmission range. Consequently, if a SOAP request fails (e.g., because of mobility), the likelihood is high that the next broadcast read from the socket is from the user, who already left the transmission range. The chains of failed requests add significant delays to the message exchange. This problem occurs more often as the number of broadcasts increases, e.g., more users or a shorter broadcast interval. Such an effect of "SOAP timeout" could not be determined in [2], since the application was reimplemented for a simulation environment. The reimplented version neither used a real SOAP library nor considered socket timeouts.

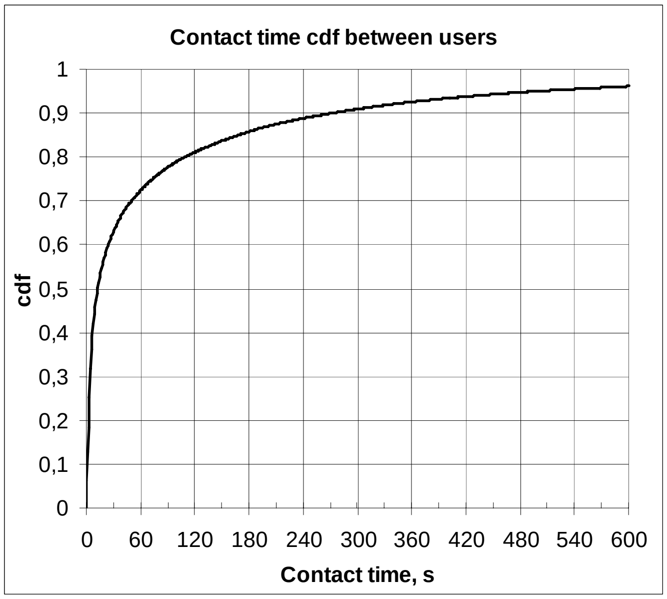


Figure 7: Cumulative distribution function of contact time between 100 mobile users

Instead of simply adjusting the timeout value, we solve the problem in a different way. We modify the "broadcast receiver", so it does not issue SOAP calls any more. It processes the received broadcasts immediately and stacks the extracted addresses ("last in, first out" paradigm). An extra thread called "synchronization scheduler" processes the stacked items and issues SOAP calls. Hence, the more recently received broadcasts are processed before others. It also performs item aging, so the addresses received more than "SOAP timeout" seconds ago are removed without processing. This reduces the lock probability. The modified implementation performs as expected (Figure 8-Figure 11).

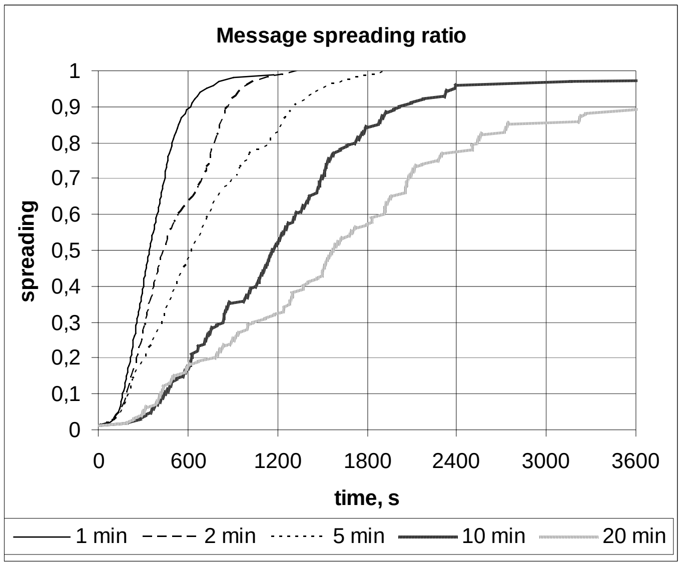


Figure 8: Message spreading ratio between 100 mobile users for different broadcast intervals. Modified application

Figure 8 shows the message spreading time for different broadcast intervals. Clearly, more frequent broadcasts lead to faster neighbor discovery, and hence, to faster message exchange.

Obviously, the configured broadcast interval impacts device energy consumption (Figure 9). We used ns-2 to estimate the energy spent for communication (without consideration of other energy-consuming components, such as CPU, display, etc.). The results are presented both in Joules and in percent of the capacity of an iPAQ battery.

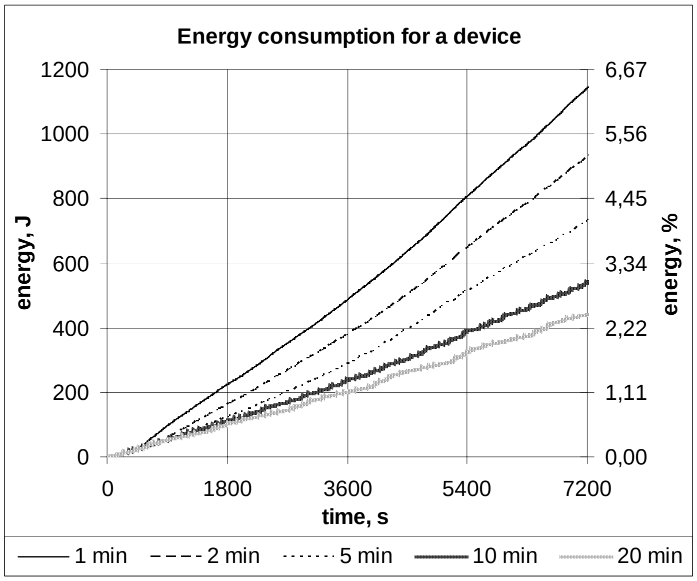


Figure 9: Average energy consumption for a mobile device. Modified application, 100 mobile users, different broadcast intervals

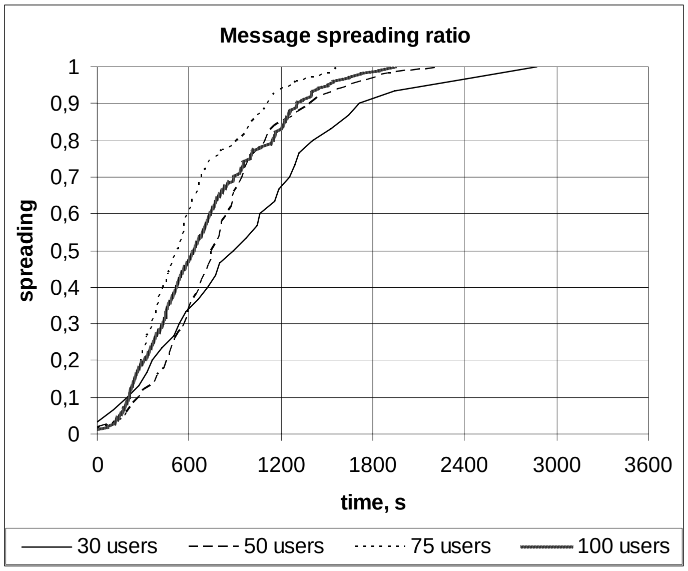


Figure 10: Message spreading ratio for different numbers of mobile users. Broadcast interval is 5 minutes. Modified application

The most essential conclusion from the results is that a message reaches 90% of users even with 20 min broadcast interval. However, an interval of 1 min requires more than 6% of device energy spent only on communication in 2 hours, while 20 min interval requires about 2.5%. Clearly, the interval duration must be configured taking the desired message spreading delay and energy consumption into account.

The message spreading time is variable in scenarios with different numbers of mobile users (Figure 10). As the number of users increases from 30 to 75, we get faster message spreading, as expected. Increasing the number of users further leads to a slower spreading. According to application logs, the number of failed SOAP requests grows due to collisions in a denser network. This also makes a device spend more energy on retransmissions (Figure 11).

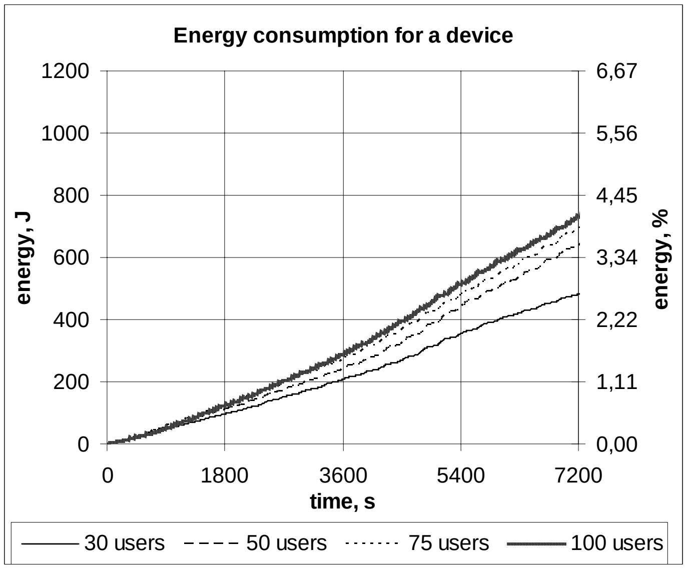


Figure 11: Average energy consumption for a mobile device. Modified application, 5 minutes broadcast interval, different number of mobile users

VIII. RELATED WORK

To the best of our knowledge, we are the first to simulate a real MANET application with such a detailed platform in a city scenario. Most of MANET evaluations use either synthetic traffic such as constant-bit rate, or reimplement applications for a specific simulation environment. As we showed in this paper, this may hide some implementation-specific aspects of an application.

As for mobility modeling, most of MANET papers use the random waypoint mobility model described in [5]. The model neglects user trip sequences and movement area constraints. Hence, it is unrealistic for city scenarios. The MANET simulation tools [4], [12], [23] commonly use this model primarily due to its simplicity. In [18], the authors simulate mobility according to real-world traces. However, they use a simple radio propagation model, which assumes an obstacle-free area and a free line-of-sight between all communicating partners.

Detailed statistics in [19] about publications at premium conferences in the field proves that the papers with such simple radio propagation models outnumber others significantly. These models poorly reflect radio wave propagation in a typical outdoor environment, such as city, in which buildings significantly affect the communication between mobile devices. The mentioned MANET simulation tools offer only the simple models. It is has been shown that the usage of more realistic radio propagation models changes simulation results considerably [24].

Also transmission errors are reflected seldom in MANET simulations. Like this paper, [22] and [30] rely on measurements from [17]. However, they still use the simple propagation models, which poorly reflect city scenarios.

As for live network traffic capturing, the ns-2 offers only passive capturing using the Berkeley Packet Filter. The implementation from [20] supports traffic capturing and injection to and from UML virtual machines. However, all the processes must run on the local host. The code also requires root privileges. Our approach support traffic capturing from UML machines running on remote hosts. Since we use only UNIX domain sockets for communication with virtual hosts, our code does not require super-user privileges.

IX. Conclusion

In this paper, we extended ns-2 for the modeling of MANET applications in city scenarios. We applied more realistic mobility and wireless transmission models. They consider movement area constraints and communication obstacles, which are taken from a digital map. We also integrated real MANET applications and implementations of network protocols.

We used the described platform for evaluating the performance of a real application (Usenet-on-the-Fly) in Stuttgart downtown. We prepared the corresponding simulation scenario. We showed that the described platform helps identifying application problems before deployment. For example, we noticed the negative effect of message buffering upon device discovery and improved the implementation. The presented simulation results give an impression of the performance of the application and energy consumption.

References

[1] AWE Communications Home Page. Available at http://www.awe-communications.com

[2] C. Becker, M. Bauer, and J. Hähner, "Usenet-on-the-fly: supporting locality of information in spontaneous networking environments," in *Proceedings of CSCW 2002 Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments*, New Orleans, USA, 2002.

[3] M. Ben-Akiva, M. Bierlaire, "Discrete choice methods and their applications to short-term travel decisions," in R. Hall (ed.), *Handbook of Transportation Science*, International Series in Operations Research and Management Science, Vol. 23, Kluwer, 1999.

[4] 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," *IEEE Computer*, Vol. 33, No. 5, May 2000.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. 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*, Dallas, TX, October 1998.

[6] CarTalk 2000 Project Home Page. Available at http://www.cartalk2000.net

[7] J. Dike, "A user-mode port of the Linux kernel", in *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta, USA, October 2000.

[8] H. Friis, "A note on a simple transmission formula," in *Proceedings of IRE*, Vol. 41, 1946.

[9] Geographic Data Files (GDF) Home Page. Available at http://www.ertico.com/en/links/links/gdf_-_geographic_data_files.htm

[10] Geometry Markup Language (GML) 2.0, OpenGIS® Implementation Specification, OGC Document Number: 01-029, 20 February 2001. Available at http://www.opengis.net/gml/01-029/GML2.html

[11] J. Gibson, *The Communications Handbook*, CRC Press, ISBN: 0849383498, 1997.

[12] GloMoSim: Global Mobile Information Systems Simulation Library. Available at http://pcl.cs.ucla.edu/projects/glomosim

[13] D. Helbing, A. Hennecke, V. Shvetsov, and M. Treiber, "Micro- and Macrosimulation of Freeway Traffic," *Mathematical and Computer Modelling*, Vol. 35, 2002.

[14] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review* E 51, 1995.

[15] R. Hoppe, G. Wölfle, and F. Landstorfer, "Fast 3D Ray Tracing for the Planning of Microcells by Intelligent Preprocessing of the Database," in *Proceedings of the 3rd European Personal and Mobile Communications Conference (EPMCC) 1999*, Paris, France, March 1999.

[16] M. Horton, "Standard for interchange of USENET messages," RFC 850, June 1983. Available at http://www.faqs.org/rfcs/rfc850.html

[17] *Intersil® HFA3861B Direct Sequence Spread Spectrum Baseband Processor Data Sheet*. File Number 4816, January 2000.

[18] J. Jetcheva, Y.-C. Hu, S. PalChaudhuri, A. Saha, and D. Johnson, "Design and Evaluation of a Metropolitan Area Multitier Wireless Ad Hoc Network Architecture," in *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems & Applications*, Monterey, USA, October 2003.

[19] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions", in *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, Venice, Italy, October, 2004.

[20] D. Mahrenholz and S. Ivanov, "Adjusting the ns-2 Emulation Mode to a Live Network," in *Proceedings of Communication in Distributed Systems 2005 (KiVS'05)*, Kaiserslautern, Germany, March 2005.

[21] E. I. Pas, "Recent Advances in Activity-Based Travel Demand Modeling," in *Proceedings of Activity-Based Travel Forecasting Conference*, New Orleans, USA, June 1996.

[22] J. del Prado and S. Choi, "Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement," in *Proceedings of the 38th annual IEEE International Conference on Communications (ICC)*, Anchorage, USA, May 2003.

[23] G. Riley, "The Georgia Tech Network Simulator," in *Proceedings of ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools'03)*, Germany, August 2003.

[24] I. Stepanov, D. Herrscher, and K. Rothermel, "On the Impact of Radio Propagation Models on MANET Simulation Results," in the *Proceedings of 7th IFIP International Conference on Mobile and Wireless Communications Networks (MWCN 2005)*, Marrakech, Morocco, September 2005.

[25] I. Stepanov, P. Marron, and K. Rothermel, "Mobility Modeling of Outdoor Scenarios for MANETs," in *Proceedings of the 38th Annual Simulation Symposium (ANSS'38)*, San Diego, USA, April 2005.

[26] M. Takai, J. Martin, and R. Bagrodia, "Effects of wireless physical layer modeling in mobile ad hoc networks", in *Proceedings of the 2nd ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc'01)*, Long Beach, USA, October 2001.

[27] U. Weidmann, "Transporttechnik der Fussgaenger," *Schriftenreihe des Instituts für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau*, Nr. 90, ETH, Zuerich, 1993.

[28] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11b, 1999.

[29] G. Wölfle, R. Hoppe, and F. Landstorfer, "A Fast and Enhanced Ray Optical Propagation Model for Indoor and Urban Scenarios, Based on an Intelligent Preprocessing of the Database," in *Proceedings of the*

10th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Osaka, Japan, September 1999.

[30] X. Wu and A. Ananda, “Link Characteristics Estimation For IEEE 802.11 DCF Based WLAN,” in *Proceedings of the 29th Annual IEEE Conference on Local Computer Networks (LCN 2004)*, Tampa, USA, November 2004.