[MHD+07] *Parag Mogre, Matthias Hollick, Nico d'Heureuse, Hans Werner Heckel, Tronje Krop, Ralf Steinmetz;* **A Graph-based Simple Mobility Model;** 4. Workshop zu Mobilen Ad hoc Netzen (WMAN'07) im Rahmen der 15. ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS'07), Bern, März 2007, S. 421-432.

A Graph-based Simple Mobility Model

Parag S. Mogre, Matthias Hollick, Nico d'Heureuse, Hans Werner Heckel, Tronje Krop, Ralf Steinmetz

Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Merckstr. 25, 64283 Darmstadt, Germany
{pmogre|mhollick}@KOM.tu-darmstadt.de

Abstract Simulation of mobile ad hoc networks requires mobility models to calculate the movement of the mobile nodes being simulated. Due to their simplicity and ease of instantiation, random mobility models are very popular in the research community. However, these models cannot easily be used to realistically model the movement of nodes in an urban scenario (i.e., terminals carried by pedestrians or in cars). Many of the existing mobility models for urban environments are based on extensive empirical data collection and, hence, are quite complex and inflexible. We adapted existing random mobility models to model movement within a city more realistically, while keeping the virtue and simplicity of random models. Our new models are graph-based, where the graph represents the streets of a city. Realistic speeds are assigned to the nodes, and, as in real life, nodes can travel together in a group. Thus, the work presented in this paper enables the use of simple and realistic random mobility models in a city scenario. Additionally, our models are designed to avoid the common pitfalls of existing random mobility models.

1 Introduction and Motivation

Simulations are a fundamental tool for researchers in the domain of mobile ad hoc networks, e.g., to test the performance of newly developed protocols. The aim of a simulation study is to obtain insights on how the system might react in practice, to see what problems might occur, and where improvement is needed. This allows researchers to avoid the costs involved for testing new protocols in real testbeds or reality. The simulation must take into account all relevant parameters and represent the real world as precisely as required. Simulations usually model a limited area of investigation. Important parameters include the size of this simulation area, the number of simulated nodes, where and how the nodes are placed (e.g., according to a random distribution), how nodes communicate with each other, and how they move. This paper focuses on the latter: realistic node movement within urban areas.

Existing, well known, models to describe node movement include *(standard) random walk* and *(standard) random waypoint*, both of which are memoryless, and models with memory such as the Gauss-Markov mobility model. Random waypoint and random walk are modeled to capture the behavior of individual users. However, the standard versions of these models are not bound to streets. Hence, nodes can move around freely in the simulation area. In general, most existing random models have not been designed for city environments, but consider the simulation area to be homogeneous, i.e., they neglect streets, buildings, etc. Our goal is to capture random city scenarios, i.e., a user (e.g., a taxi-driver) is traveling through random streets of the city, with no obvious destination, going back and forth, and pausing every now and then.

There have been attempts in the literature to develop realistic mobility models for cities. Typically, these models are graph-based mobility models or obstacle mobility models, which restrict the movement of the mobile nodes to the streets within the simulation area or avoid obstacles such as buildings. Unfortunately, many realistic mobility require extensive empirical data collection, and in general are quite complex. The simpler stochastic mobility models often exhibit unrealistic node density distributions. Our goal is to achieve a realistic mobility model that is as simple and efficient as existing random models.

Our *graph-based mobility models* fill the gap and limit random movement to a graph. In addition we design an efficient mechanism for controlling the density distribution of nodes over the simulation area (e.g., uniform distribution of nodes on streets). This involves the design of a suitable mechanism for choosing the initial position of nodes as well as the selection of the next waypoint in our so-called graph random waypoint mobility model. Additionally, we tackle *grouped mobility*: a number of mobile nodes may which are close together and move along the same path. The radios are still independent of each other, but the group itself behaves like a single node and will follow the same mobility pattern.

Our contribution is as follows:

- We design a *graph random walk* and a *graph random-waypoint* model, which are both, realistic and simple/efficient.
- We describe an extension to allow for modeling of group mobility using our models.
- We give a detailed description of the implementation of the models in the JiST/SWANS simulation environment [1].
- We analyze our models by means of a simulation study, which also serves as a proof-of-concept to validate our design goals.

In the following, we briefly present related work and describe existing mobility models. This is followed by a detailed description of our novel graph-based mobility models. In particular, we describe the model assumptions and requirements and thoroughly describe the implementation of the models. As a proof-of-concept, we then perform an experimental analysis to obtain selected properties of our model. We conclude our work by discussing possible future work.

2 Related Work

Mobility models can be categorized in *macroscopic* and *microscopic* models. Macroscopic mobility models realistically describe the aggregated effects of mobility for large areas. Instantiation of these models is typically performed using

census data. Activity-based macroscopic models borrow the concept of trips to describe the mobile user's behavior on different scale [2,3]. For a detailed discussion of realistic macroscopic mobility models for metropolitan areas see [4]. The process of instantiation of the macroscopic mobility models is, however, rather complex and involves empirical data collection.

In contrast, microscopic models describe the average mobility behavior of individual entities, usually, by means of a mathematical description of the paths of these entities in the simulated area. Most synthetic or random models fall into this class [5], which includes the random walk as well as the random waypoint mobility model [6]. In the latter, each mobile node randomly selects a position in the simulation area as its destination. The node then moves toward this destination with a constant velocity chosen uniformly and randomly from $[0,V_{max}]$. The parameter V_{max} specifies the maximum speed of the mobile nodes in the simulation. On reaching the destination the mobile node may pause for a short interval while it chooses the next destination to travel to. In contrast to the random waypoint model, the nodes in the random walk model move with a randomly selected constant speed in a randomly selected direction for an interval of time. After this time interval, the individual nodes select another random direction of travel and another random speed. The main strength of these random models is the virtue and simplicity of implementation and usage. Due to their simplicity and ease of use, they are available widely in simulation platforms and have been used extensively in performance evaluations of ad hoc routing protocols.

A typical scenario where ad hoc networks (or vehicular networks) composed of mobile nodes are expected to thrive and exist in future are urban, city based scenarios. Both random walk and random waypoint models, however, describe a "flat" simulation area without forbidden or preferred zones of movement (obstacles, buildings, streets). This is one of the major points of criticism for the above mobility models. Recently, the authors in [7] proposed the random trip mobility model, which provides a generalization of random waypoint and the random walk mobility models in realistic settings. Although it is a generic framework to obtain good mobility models, it is quite complex and requires careful parameterization on part of the user. One of the simpler random mobility models that attempts to model the movement of mobile terminals in realistic scenarios such as a city can be found in [8]. The authors introduce obstacles as a means to model the movement of users in a city-like scenario. However, the choice of the position at which new nodes are to be spawned in the simulation field and the strategy of nodes to select their next destination are left open. One can find many other mobility models which intend to introduce realistic aspects to the mobility model by introducing various mobility patterns for different types of nodes [9], or by restricting mobility to certain areas, for example via graphs, see Refs. [8–11] for some proposed mobility models. These models allow a fair amount of freedom in defining the scenario (streets, buildings, node velocity, street signals, traffic congestion etc. However, the initialization of the above models is quite tedious and the protocol performance results obtained from the above simulation hold for the above scenario, but may be completely different for a different city models, street speeds, obstacle placements, etc. Thus, to obtain statistically sound and representative results it is necessary to carry out a simulation study of the protocol in a sufficiently large number of scenarios. With the complex setup required for the above models, this can be quite time consuming (both for setup of the scenario and for the actual simulation execution). Thus, these mobility models are not perfectly suited for simulation studies where one wants to test the protocol quickly in a large number of city-like scenarios.

Our work closes the above gap and provides an efficient and realistic algorithm for positioning nodes and determining the path towards the next destination. We implement a flexible framework which enables researchers to use simple random mobility models in a fairly realistic city-like scenario. The implementation also permits assigning a different mobility model to each individual node in the simulation. In addition, we address the well-known problems of the random waypoint model [12], which lead to undesired density distributions as shown in [13].

3 Proposed Graph-based Models

Our graph-based models necessitate various extensions to the existing random models. In particular, we need to devise a solution for the initial placement of the nodes on the simulation field such that the boundary conditions (nodes have to be located on the street graph only) are fulfilled. Also, the random destinations have to be chosen such that they fulfill the boundary conditions. For our solution, we have different alternatives. For the initial node placement, we can follow the node placement strategy of the standard models in a first step and then snap the node to the closest point on a street. Another possibility is to place nodes according to a random distribution on the streets at initialization time. For the choice of the random destination, we can again follow a strategy that randomly selects a position on the available streets or that choses a random destination on the simulation area and then snaps the node to the nearest street.

Without loss of generality, for the implementation of our model, we choose to place the nodes on the simulation area and then calculate the nearest point on a street. Thus, by using different (uneven) random distributions, we can easily account for different population densities in the simulated area.

3.1 Models

Graph Random Walk (GRW). In the (standard) random walk (SRW) model, a mobile node chooses a random direction and speed, and moves for either a certain time or a certain distance. It then pauses and starts over, choosing new values, uncorrelated with the last direction and speed.

In our novel graph random walk (GRW) model, a node starts on an edge of the simulated graph. It randomly selects a total distance it is willing to travel. For starting the movement, the node has two possibilities to move along the edge. As soon as it hits a vertex, one of the edges leaving the vertex is randomly

chosen. Our algorithm does implement some memory here: the node will never move back to where it came from, unless it has entered a dead-end street and has nowhere else to move to. The step size memorized is 1. When the distance chosen in the beginning has been traveled, the node stops (on the edge), pauses for a random time, and starts over.

Graph Random Waypoint (GWP). In the (standard) random waypoint model (SWP), a node selects a random point anywhere within the simulation area. It then moves to the selected destination with a randomly selected speed, following the shortest path. It then pauses and starts over. Imitating that behavior, our new GWP model selects a random destination somewhere on an edge of the graph. It then calculates the shortest path to get there. The cost of a segment, used to calculate the shortest path, is the time necessary to travel along that street segment. The mobile node follows that path until the destination is reached. It then pauses and starts over, selecting a new destination.

3.2 Implementation

The implementation was performed using the JiST/SWANS framework [1]. The JiST/SWANS documentation gives further information about the simulation API and the interfaces for handling mobility models, node locations, etc.

Random Values. To allow for flexible parameterization of our models, we decided to be as generic as possible with respect to the employed random number generator. Our proposed models make use of random values, e.g., to determine where a new node should be positioned, how fast it should travel, or how long a pause should be. To be flexible, we need to be able to incorporate different random distributions for these different tasks. While the position of a new node might be uniformly distributed over the entire simulation area, the travel speed could be Gaussian distributed with a certain variance and a mean corresponding to the speed limit on the street. We have chosen a flexible random number generator to fulfill the aforementioned requirements. The `ValueServer`-class of the commons.math-package [14] is such a generic number generator. Its default implementation allows to specify different distributions (constant, uniform, Gaussian, exponential) and their respective parameters (mean, variance), which can be specified in the simulation setup.

GraphRandomWalk (GRW). The implementation of the GRW mobility model as described in Section 3.1 is quite straight forward: the `Mobility`-interface of JiST/SWANS specifies two methods which have to be implemented: `public MobilityInfo init()` and `public void next()`.

The `init()`-method is called to initialize the mobility model. It returns an object of type `MobilityInfo` (in our case `GraphRandomWalkInfo`) which stores the information about the movement that is needed in the next movement step. This includes:

- the remaining *distance* until the node pauses for a certain time,
- the current *speed* the node is traveling with,
- the next *waypoint* (a vertex on the graph) the node is traveling to, and
- the number of movement *steps* remaining until the next waypoint is reached.

The *waypoint* is an object of type `LocationContainer`. It stores the `Location` the node started from and the `Location` the node travels to as well as the corresponding vertices of the graph. The start and end locations may be situated anywhere on the edge given by the two vertices.

The `init()`-method is called by the `MobileInterface` (usually of type `Field`) containing the node to be moved. When the method is called, the *id* and the current location of the node are passed to the method. The first step in the movement process is to snap the node to the nearest edge on the graph (since the passed location might not be on the graph). Afterwards, one of the vertices of the edge is randomly selected as the next destination. These two steps are both done by a call of `MobilityGraph.getNext(int xPosition, int yPosition)`. It returns an object of type `LocationContainer` which is stored in `GraphRandomWalkInfo` and contains a start location (the location the node is snapped to) and a destination location. This destination is the first *waypoint*. A travelling speed is generated by a `ValueServer` (specified upon construction of GraphRandomWalk) using the mean *speed* of the edge the node is currently traveling on as the mean of the random distribution. The last step of the initialization process consists of choosing a random distance after which the node pauses. This is again done by a `ValueServer` specified in the constructor.

The `next()`-method is called by the `MobileInterface` each time the node should perform or schedule its next step. The node moves along the edge until either the waypoint is reached or the random distance, chosen in `init()`, has been travelled. If a waypoint is reached, a new waypoint is generated as explained in Section 3.1 by calling `MobilityGraph.getNext(Vertex startVertex, Vertex destinationVertex)`. A new travel speed is chosen accordingly.

The distance of each step is subtracted from the random distance chosen in `init()`. If this distance finally becomes zero, the node pauses for a random time. This time is again generated by a ValueServer. After the pause, the movement process has to be restarted, which is achieved by calling `init()`.

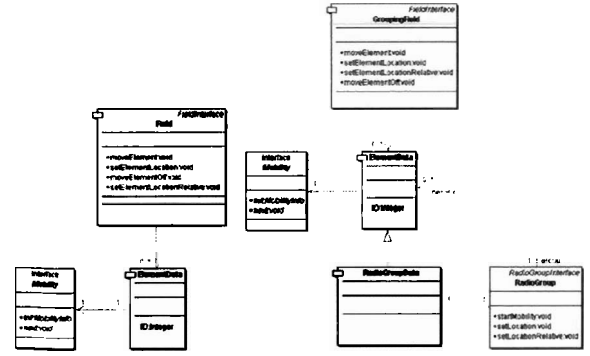
GraphRandomWaypoint (GWP). In the GWP model, the nodes follow a path, which is chosen as explained in Section 3.1. We base our implementation on the JUNG `Graph` classes (Java Universal Network/Graph, see [15]), thus, a path returned by the JUNG methods contains a `List` of JUNG `Edges`. These can not be directly used for the movement of the nodes in JiST/SWANS. Instead the `Location` of the vertices and the mean speed on the edges has to be extracted from this `List`. To represent the path in an adequate way, the class `Waypoint` is introduced. An object of this type stores a `Location`, the *speed* that should be used to travel to this `Location`, and the next `Waypoint` (which is `null` if the last `Location` was the end location of the path).

For the concrete implementation, the two methods `init(...)` and `next(...)` have to be implemented once more. For this mobility model, the information that has to be passed from one step to the next is stored in an object of type `GraphRandomWaypointInfo`, which contains:

- the current *speed* the node is traveling with,
- the *path* the node still has to travel on, and
- the number of movement *steps* remaining until the next waypoint is reached.

During the initialization, the node has to be snapped to an edge of the graph first (see section 3.2). A random destination `Location` is generated using `ValueServers`. The fastest path from the current Location to the newly generated destination (see section 3.1) can now be computed. This path is stored in the `Waypoint` object of the `GraphRandomWaypointInfo` as explained before.

A call of the `next()` method starts the movement of the node along the path. For each edge a new random speed is chosen with a mean which is equal to the speed stored in the `Waypoint` object. When the final destination is reached, the node pauses for a random time. Afterwards the movement can be restarted, which is again done by calling the `init()` method.



(a) UML class-diagram for `Field` (b) UML class-diagram for `GroupingField`

Figure 1: UML class-diagramms

Grouped Mobility The `GroupingField` class adds the possibility of grouped nodes to the `Field` class. This means that the nodes can move in a parent-child-relationship, where the movement of the parent affects the movement of the children. Fig. 1a shows a class-diagram of the existing `Field` class as it was already present in JiST/SWANS. Fig. 1b shows the extended structure of the new `GroupingField` class. The `RadioGroupData` class is an extension of the `ElementData` class and holds an additional reference to an object of the `RadioGroup` class. This class holds the set of `ElementData` objects that belong to this group. The movement of the elements is now performed as follows:

1. A `Mobility` object calls the `moveElement(id, location)` method of the `GroupingField`.
2. Using the *id*, the corresponding `ElementData` object is retrieved from `Field`.
3. The `Location` of this object is set to the new `Location`.
4. If the element is an instance of `RadioGroupData`, the `move(...)` method of the `RadioGroup` is called.
5. Knowing its old `Location` as well as the new `Location`, the `RadioGroup` now calculates its relative movement.
6. The `RadioGroup` then moves its members by the calculated relative movement by calling the `setElementLocationRelative(...)` method of the `GroupingField` for each group member.
7. If the member is a group itself, the process is recursively restarted from 3.

4 Experimental Evaluation

We evaluated our models presented in Section 3 in different scenarios. Goal of our analysis is to address the following issues:

- Node distribution: does our GWP model exhibit a clustering effect similar to the one observed in the SWP model [13]?
- Speed decay: do our models exhibit a decay of the mean speed over simulation time as observed in the standard models [12] and how does the mean pause time of the nodes affect the average speed observed?

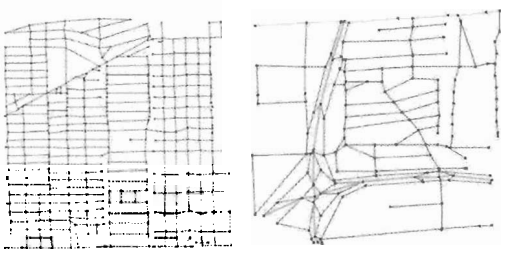
We evaluated the mobility models on two different road scenarios. Scenario I, which is shown in Fig. 2a represents a part of a city (2400m*2400m) with a dense street network where all the streets are of equal type, meaning that they all have the same speed limit of $40km/h$. Scenario II, shown in Fig. 2b, has a greater variety of streets (1900m*1900m, including highways with a speed limit of $70km/h$) and is not as dense as Scenario I. The road data was taken from the public Tiger/Line-Database [16]. Each of the results presented in this section is the average of 20 simulations with 100 nodes each. For the GRW model, the travel distance was Gaussian distributed with mean $\mu_d = 5km$ and a variance of $\sigma_d = 1.5km$.

4.1 Node Distribution

Fig. 3 shows the utilization of the streets of our scenarios after a simulated time of 48 hours, where utilization is measured as the total time the street is used.

The travel speed was Gaussian distributed, with the mean $\mu_{s,i}$ of street i set to the speed limit of street i. The variance was set to $\sigma_{s,i} = 0.1\mu_{s,i}$. For the SWP model, a mean speed of $\mu_s = 40km/h$ was used, which is the same as the speed limit of the streets in Scenario I. A variance of $\sigma_s = 0.1 \cdot \mu_s$ was used for the SWP simulation, too. The pause time of the nodes was set to zero for all models.

In the GRW mobility simulation, the utilization is uniformly distributed on all streets in Scenario I (see Fig. 3, left), as it is for a standard random walk (SRW) model [5]. In Scenario II (see Fig. 4, left), one observes that the utilization of the highways is lower than the one of the normal streets. This is due to the fact that a node chooses its next streets independently of the street speed limits. Since the speed on the highways is higher than the speed on the normal streets, the mean time a node travels on a part of a highway is lower than the mean time spent on a part of a normal street of the same length. In the GWP scenario, a clustering of the nodes is clearly visible (see Fig. 3 and Fig. 4, right). While in the SWP model the node density is higher at the center, in the GWP model, clustering does not necessarily occur in the center. In GWP the nodes check for the fastest path from one location to another. The utilization of a street depends on its mean speed and on the shape of the street grid itself. Due to the nearly uniform distribution of streets in Scenario I (see Fig. 3, right), the streets are used with similar probability. But as in the SWP model, most nodes want to travel through the center. This is why there is a higher utilization of the streets in the center of the simulation area. In Scenario II (see Fig. 4, right) most fastest paths from one end to the other end of the simulation area use the highways. These streets are more heavily loaded than the streets in the center.



(a) Scenario I (b) Scenario II

Figure 2: Street scenarios used for evaluation.

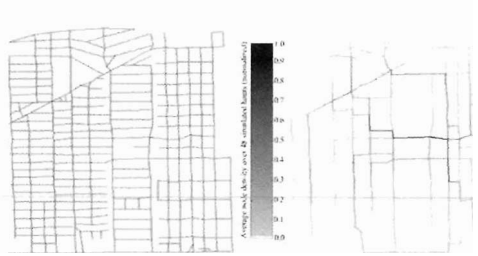


Figure 3: Node distribution in Scenario I using *graph random walk* (GRW, left) vs. *graph random waypoint* (GWP, right) mobility model.

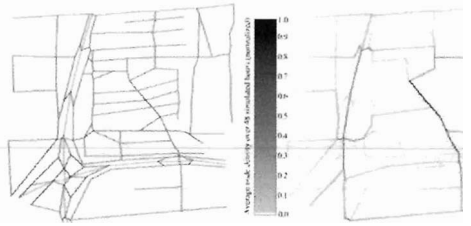
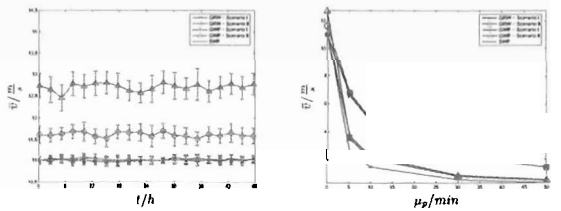


Figure 4: Node distribution in Scenario II using *graph random walk* (GRW, left) vs. *graph random waypoint* (GWP, right) mobility model.



(a) Mean speed over time (b) Mean speed vs. pausetime

Figure 5: (a)Mean speed $\bar{v}$ over time t and (b) mean speed $\bar{v}$ vs. pausetime μ_p

4.2 Speed Decay

For the random waypoint model it has been shown that the mean speed of the nodes decays over time, since more and more nodes get stuck on long paths with low speeds [12]. Fig. 5 (a) shows the instantaneous mean speed $\bar{v}$ of the nodes over a simulated time of 48 hours using the same speed settings as in Section 4.1. The instantaneous mean speed is defined as

$$\bar{v}(t) = \frac{\sum_{i=1}^{N} v_i(t)}{N}$$

with N being the total number of nodes and $v_i(t)$ the speed of node i at time t. Since there are streets with a speed limit of more than $40km/h$ in Scenario II, the overall mean speeds for Scenario II are higher then the corresponding speeds for Scenario I. Since the highways in Scenario II are used more often for GWP than for GRW, the mean speed in Scenario II for GWP is higher than the mean speed for GRW. Using our settings, no speed decay can be observed within the simulated time. This is due to the fact that the speeds of the nodes are not chosen from a uniform distribution but from a Gaussian distribution with a rather small variance. Fig. 5 (b) shows the influence of the mean pause time μ_p on the overall mean speed $\bar{v}$. The pause time being Gaussian distributed with a mean of μ_p varying from 0 to 70 minutes and a variance of $\sigma_p = 0.3 \cdot \mu_p$. As expected, the mean speed drops with increasing pause time. In our simulation, the speeds for GWP dropped faster than the speeds of GRW. This is because the distribution of the travel distances used in GRW is not equal to the path length distribution for GWP: in GWP the nodes pause more often, thus leading to a lower mean speed.

5 Conclusion

In our work, we introduced two novel graph-based schemes for simple and efficient modeling of node mobility for use in simulation environments: *graph random walk* and *graph random waypoint*. In combination with the possibility to group nodes, this presents a powerful tool for realistic simulation of real-world movement, while keeping the instantiation process of the model rather simple and intuitive. Our models are based on the existing standard random walk and random waypoint models. In particular, we increase the realism of these standard models by limiting node movement to predefined streets. At the same time, we address well-known problems of the standard models such as non-uniform node-distribution or speed decay over time. As a proof-of-concept, we implemented the developed models using the JiST/SWANS simulation environment. We performed a performance evaluation in which our models have shown to be adaptable to various kinds of city topologies and produced meaningful results. Apart from being apt for different topologies, our model as well as our implementation approach was designed to allow for flexibility, e.g., by incorporating different random distributions. The current work opens up several promising avenues for further research. Care should be taken that the quest for realism does not lead to too complicated models, which on the one hand are highly complex (also in terms of computational complexity), and on the other hand do not aid the better understanding of ad hoc protocol behaviour in real networks.

References

1. "JiST/ SWANS user guides," http://jist.ece.cornell.edu/.
2. J. Scourias and T. Kunz, "Activity-based Mobility Modeling: Realistic Evaluation of Location Management Schemes for Cellular Networks," in *Proceedings of WCNC 1999*, September 1999, pp. 296–300.
3. D. Lam, D. C. Cox, and J. Wilson, "Teletraffic Modeling for Personal Communication Services," *IEEE Communications Magazine*, vol. 35, no. 2, pp. 79–87, February 1997.
4. M. Hollick, T. Krop, J. Schmitt, H.-P. Huth, and R. Steinmetz, "Modeling Mobility and Workload for Wireless Metropolitan Area Networks," *Computer Communications*, vol. 27, pp. 751–761, Mai 2004.
5. T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.
6. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparision of Multi-hop Wireless Ad hoc Routing Protocols," in *Proceedings of MobiCom 1998*, 1998, pp. 85–97.
7. J. L. Boudec and M. Vojnovic, "Perfect Simulation and Stationarity of a Class of Mobility Models," in *Proceedings of INFOCOM 2005*, July 2005, pp. 2743–2754.
8. A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri, "Towards Realistic Mobility Models for Mobile Ad hoc Networks," in *Proceedings of MobiCom 2003*, 2003, pp. 217–229.
9. G. Lu, G. Manson, and D. Belis, "Enhancing Routing Performance for Inter-Vehicle Communication in City Environment," in *Proceedings of $PM^2HW^2N'06$*, October 2006, pp. 82–89.
10. J. Tian, J. Hähner, C. Becker, I. Stepanov, and K. Rothermel, "Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation," in *Annual Simulation Symposium (SS'02)*. IEEE Computer Society, 2002.
11. D. R. Choffnes and F. E. Bustamante, "An Integrated Mobility and Traffic Model for Vehicular Wireless Networks," in *Proceedings of VANET'05*. ACM Press, September 2005, pp. 69–78.
12. J. Yoon, M. Liu, and B. D. Noble, "Random Waypoint Considered Harmful," in *Proceedings of INFOCOM 2003*, vol. 2, April 2003, pp. 1312–1321.
13. C. Bettstetter, H. Hartenstein, and X. Perez-Costa, "Stochastic Properties of the Random Waypoint Mobility Model," *ACM/Kluwer Wireless Networks, Special Issue on Modeling and Analysis of Mobile Networks*, vol. 10, no. 5, September 2004.
14. "Commons-Math API," http://jakarta.apache.org/commons/math.
15. J. O'Madadhain, D. Fisher, S. White, and Y.-B. Boey, "The JUNG (Java Universal Network Graph) Framework," TR UCI-ICS 03-17, University of California, Irvine, Tech. Rep., October 2003, http://www.datalab.uci.edu/papers/JUNG_tech_report.html.
16. "TIGER Topologically Integrated Geographic Encoding and Referencing system, U.S. Census Bureau," http://www.census.gov/geo/www/tiger.