# MobiREAL Simulator
# – Evaluating MANET Applications in Real Environments –

Kazuki Konishi, Kumiko Maeda, Kazuki Sato, Akiko Yamasaki,
Hirozumi Yamaguchi, Keiichi Yasumoto[†], Teruo Higashino

Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyamacho, Toyonaka, Osaka 560-8531, Japan
{k-konisi, k-maeda, kz-satou, a-yamasaki, h-yamagu, higashino}@ist.osaka-u.ac.jp
[†] Graduate School of Information Science, Nara Institute of Science and Technology
yasumoto@is.naist.jp

## Abstract

*In this paper, we propose a probabilistic rule-based model to describe behavior of mobile nodes for accurately evaluating performance of MANET applications. The proposed model allows us to describe how mobile nodes change their destinations, routes and speeds/directions based on their positions, surroundings (e.g. neighboring nodes), information obtained from applications, and so on. We have designed and developed a network simulator called MobiREAL based on the proposed methodology. Through experiments, we show importance to simulate MANET applications in realistic environments.*

## 1 Introduction

Recently, in simulation of MANET (Mobile Ad-hoc Networks), reality of simulation especially reality of mobility modeling is considered to be a very important factor and several research groups have made efforts to provide realistic mobility models of nodes [2, 3].

Ref. [5] introduces obstacles and realistic pathways between them in a simulation space. Ref. [4] presents WWP (Weighted Way Point) model where a set of crowded regions is defined. Given a distribution of pause times of nodes for each region, it uses a Markov model to model node's movement between these regions. These techniques focus on macroscopic mobility, which is useful for reproducing realistic mobility of nodes. However, they do not consider microscopic behavior, *i.e.* how each node changes its behavior dynamically, depending on its surroundings, information obtained from network systems, time and so on.

In this paper, we present a new method to model and simulate (near) real movement and behavior of nodes in mobile ad hoc network simulation. In the proposed method, we adopt a rule-based model to describe movement/behavior patterns of mobile nodes. In our model, mobile nodes are classified into multiple groups depending on their behavior patterns. For each group of mobile nodes, a rule-based description is specified where dynamic and realistic behavior of nodes such as visiting favorite shops in shopping, avoiding congested routes and receiving/distributing publicity information through network systems can be specified. Then node objects are instantiated with the initial parameters according to a simulation scenario. Based on the proposed methodology, we have developed a network simulator called *MobiREAL*. Through case studies, we show the effectiveness of our framework.

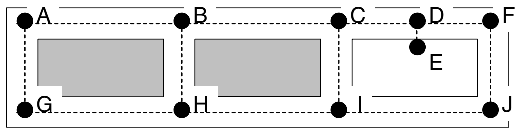


**Figure 1. Modeling simulation field.**

## 2 Our Mobility Model

### 2.1 Simulation Field

Simulation fields are modeled as follows. Polygons are used to represent buildings, parks and so on. Each polygon has an attribute that indicates whether the region is accessible or not and whether radio signal penetrates its boundary or not. If it is accessible, it may have some points on its boundary to represent entrances of the region. We allow to specify a set of points on streets or intersections and connects them to represent road structures (see Fig. 1).

### 2.2 Mobile Node Behavior in CPE Model

Our policy to specify the behavior of nodes (mainly pedestrians in city sections with hand-held wireless devices) is following. (i) We first classify the types of mobile nodes by their behavior. (ii) Then for each type of mobile nodes,

**[CPE Description]**

| | Condition | Prob. | Action |
|---|---|---|---|
| E1 | crowded(P,V,E)<br>front region is crowded | norm_dist(5.0min, ±3.0min) | dst.r=detour_path(P,E,dst.p); V=norm(P,V,E,dst);<br>detour crowded region |
| E2 | late(P,V,T,dst)<br>in time if hurries up | 1.00 | V=fast(P,V,E,dst);<br>speed up |
| E3 | miss(P,V,T,dst)<br>not in time even if hurries up | 0.20 | V=fast(P,V,E,dst); dst.t+=10min;<br>add 10 minutes to expected arrive time and speed up |
| E4 | miss(P,V,T,dst)<br>not in time even if hurries up | 1.00 | dst=pop(Dlist); dst.r=shortest_path(P,dst.p); V=norm(P,V,E,dst);<br>give up going to the current destination and go to the next destination |
| E5 | receive_from_net(AO,new_dest)<br>acquiring information from the system | 0.50 | put(new_dest,Dlist); V=norm(P,V,E,dst);<br>add a destination obtained from the system to a destination list |
| E6 | ¬staying ∧ ¬overstaying ∧ reach(P,dst.p)<br>arrived at destination | 1.00 | sst=T; staying=true; V=0;<br>record arrival time and set staying true |
| E7 | staying ∧ (T−sst)≥dst.s<br><br>expected staying time has passed | 0.80 | staying=false; dst=pop(Dlist); dst.r=shortest_path(P,dst.p); V=norm(P,V,E,dst);<br>go to the next destination |
| E8 | staying ∧ (T−sst)≥dst.s<br>expected staying time has passed | 1.00 | staying=false; overstaying=true; dst.s+=rand(5*MIN, 10*MIN);<br>extend staying time |
| E9 | overstaying ∧ (T−sst)≥ dst.s<br><br>expected staying time has passed during overstaying | 1.00 | send_to_net(AI,dst.p); overstaying=false; dst=pop(Dlist); dst.r=shortest_path(P,dst.p); V=norm(P,V,E,dst);<br>input the present destination to the system and go to the next destination |

**[Internal Variables]**

dst.p = E  dst.r = {H, I, C, D, E}  dst.t = 3:10  dst.s = 30 min  Dlist = {dst2, dst3, ...}  staying, overstaying = false  sst = 0

**Figure 2. CPE modeling of pedestrian behavior in city section.**

we specify their behavior in CPE (Condition-Probability-Event) model introduced later. Each CPE description corresponds to a "class" of mobile node objects that have similar behavior. Thus, the destination and route of each mobile node object, which moves based on a CPE description, are represented as variables in the CPE description. (iii) Finally, we describe a simulation scenario that instantiates the mobile node objects with specifying their initial positions, destinations and so on.

We introduce *Condition-Probability-Event* model (CPE model) to describe the behavior of mobile nodes. We present our CPE modeling of pedestrian behavior in city sections in Fig. 2. The CPE model consists of a list of rules, and internal and external variables. The external variables can be accessed (and updated) from the outside of the CPE model. They include simulation clock $T$, surroundings information $E$ of the node, output data $AO$ from the network system to the node, input data $AI$ of the node to the network system, current position $P$ and velocity vector $V$ of the node. Each rule is a tuple of a *condition*, a *probability* and an *action*. Logical formula using the variables can be specified as a condition, and a value [0,1] or a probability function can be specified as a probability. As an action, a set of substitution statements which update values of the variables is specified. The model is executed as follows. The simulation clock $T$ is incremented automatically from the outside of the CPE model, and for each increment, an executable rule is searched from the top of the list, and a rule that satisfies its condition and probability is selected to be executed. This search ends if it executes a rule, otherwise reaches the bottom of the list.

In Fig. 2, an internal variable $dst$ is a structure which represents a destination and has the following members; $p$ (destination point), $r$ (a sequence of points (route) to reach the destination point $p$), $t$ (estimated arrival time) and $s$ (stay time at the destination (pause time)). $Dlist$ keeps a list of $dst$'s which will be visited later. Functions in the rules are pre-defined or user-defined and we omit the details. In Rule $E1$, the value of logical formula $crowded$ becomes true if the node finds crowded region ahead. Then the rule may be selected depending on the probability. In this rule, the probability function, which depends on simulation time, returns a probability value so that the rule can be selected every 5 minutes on average, with maximum 3 minutes variations (these variations follow a normal distribution). Therefore, once the node executes $E1$, this rule is not selected again until 5 minutes passes on average. This prevents too much (unnecessary) frequent execution of the same rule. If the rule is selected, $dst.r$, the route to the destination $dst.p$, is re-calculated by $detour\_path$ to detour the region. If $E1$ is not selected, then $E2$ is checked. The value of the condition of $E2$ becomes true if the node recognizes that it cannot arrive at the destination until the estimated arrival time $dst.t$ but may be able to arrive in time with a higher speed. In this case, it changes the velocity vector $V$ appropriately.

We note that the vector calculation functions $slow$, $norm$ and $fast$ in this example calculate the velocity to the nearest point in route $dst.r$ from $P$. For example, in Fig. 1, if a node sets point $E$ to $dst.p$ (destination) and if it resides on the street between $G$ and $H$, $dst.r$ keeps the route $\{H,I,C,D,E\}$ (this is calculated by function $shortest\_path$ in some rules). In this case, the vector calculation function uses the current position $P$ of the node and the nearest point $H$ in $dst.r$ to calculate vector $V$. We also note that people sometimes walk zig-zag to avoid others approaching, or to avoid local small obstacles. By such collision avoidance among neighboring persons, velocity vectors may deviate temporarily, and after a while it is regulated. By realizing such location deviation, we present some library functions that can be used in the velocity vector calculation functions.

We have other rules up to $E9$ in Fig. 2. For example, rules $E5$ and $E9$ represent the behavior to use the network system. Here we have assumed that the network system is implemented on each node's terminal with a short range wireless device such as IEEE802.11, and these terminals communicate with each other in an ad hoc manner to exchange the stored information such as recommended shops and ads. Rule $E5$ represents that if the node receives information (*e.g.* recommended shop information nearby) from its terminal and if the node is interested in the point (probability is 0.50), then it adds the point to its list $Dlist$ of destinations. Explanations of other rules are described in Fig. 2.

### 2.3 Simulation Scenario

For each type of mobile nodes, a CPE model is specified. In the simulation scenario, we give the initial values of external variables $P$ (position) and $V$ (vector) and internal variables (*e.g.* a set $Dlist$ of destination) to instantiate objects (entities) of the CPE models. For example, the objects of "customer" are generated from the corresponding CPE model. We can determine $V$, $P$ and $Dlist$ and generation ratio of nodes with some randomized factors as follows.

```
V = RandomV( 1.0, 1.8);
if ( rand() <= 0.30)  P = A;
else  P = G;
Dlist.add( dst(H,T+req_time(T,H),30min));
if (normal_dist(15sec,1sec))
  customer.gen(V,P,Dlist);
```

## 3 MobiREAL Simulator Overview

MobiREAL simulator is composed of two parts called *MobiREAL behavior simulator* which simulates mobile nodes' behavior and *MobiREAL network simulator* which simulates data exchange among mobile nodes. The behavior and network simulators are two independent programs that periodically exchange necessary data through a TCP channel.

For our MobiREAL behavior simulator, each simulator user must give (i) a simulation field model, (ii) a set of CPE models of mobile nodes and (iii) a simulation scenario for generating mobile node objects. These are translated into the corresponding C++ classes using pre-defined library functions. For MobiREAL network simulator, the simulator user must also give C++ simulator codes of network applications and user-specific protocols used in the simulation. When these two simulators are initialized, they hold the simulation field and node objects independently. The behavior simulator calculates the latest positions, directions and speeds of nodes, and send them to the network simulator. Node objects in the network simulator hold their positions, directions and speeds according to received data.

We have extended the network simulator GTNetS [6] developed in Georgia Institute of Technology so that dynamic node generation and deletion are supported and node positions, speeds and directions can be updated from the behavior simulator. For simulation of network and MAC layer protocols, our MobiREAL network simulator relies on GTNetS where DSR, AODV and NVR (wireless form of Nix-Vector routing) are supported in the network layer and the standard IEEE802.11 DCF (CSMA/CA) with RTS/CTS is supported in the MAC layer. We have enhanced the signal propagation model in the original GTNetS so that obstacles are taken into account. It has a similar propagation model to Ref. [5] where ground reflection is considered as zero propagation and line-of-sight is basically considered.

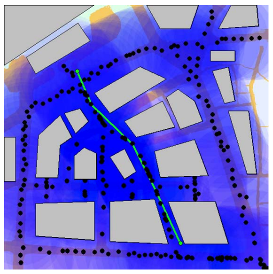
MobiREAL also has an animator which visualizes packet propagation, logical topology of MANETs, node positions/movements and so on. It works on the Microsoft Windows platforms, and has been developed using DirectX 9. In the MobiREAL web page [1] the graphical demos and snapshots are provided.

## 4 Experimental Results

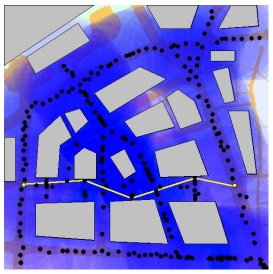
In order to validate the efficiency of our framework, we present performance evaluation of typical MANET applications using MobiREAL. For the evaluation, we have modeled a real 500m$\times$500m region including buildings and pedestrians with typical behavior patterns in downtown Osaka city.

We consider end-to-end real-time communication on MANETs as an example. We consider the following situation. In this application, we focus on the performance of the routing protocol. We will show that the performance of the routing protocol such as the number of route breaks, are largely affected by the locations of end users.

In our target field of downtown Osaka, we have big streets connecting the north and south areas. Therefore, several major convective flows of mobile nodes were formed from north to south and vice versa (this is a real situation in downtown Osaka). Obviously, if an end-to-end path for packet transmission is established from north to south (or



(a) scenario $A$ (N - S)



(b) scenario $B$ (E - W)

| | |
|---|---|
| Speed of nodes | 2 m/s |
| Average density | 0.00048 person/m$^2$ |
| Application | 10Kbps CBR traffic (between the two users) |
| Routing protocol | DSR |
| MAC layer protocol | IEEE 802.11 DCF (CSMA/CA with RTS/CTS) |
| Radio range | 100 m |
| Simulation time | 720 sec (communication was done by UDP from time 240 to time 720) |

(c) behavior/network simulator settings

**Figure 3. Experimental settings of end-to-end real-time communication.**

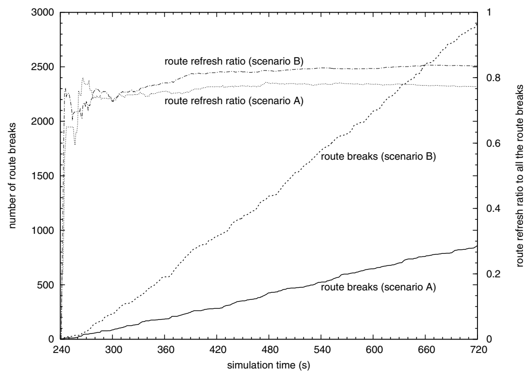
vice versa), the path follows pedestrian flows on a street and is expected to be more stable than that established in east and west direction.

According to this observation, we have prepared two scenarios $A$ and $B$. These scenarios are different only in the locations of end users (application users) who do not move during the simulation. In scenario $A$, two end users are in the north and south areas, respectively, while in scenario $B$ two end users are in the east and west areas, respectively. In both scenarios, the physical distances between the application users are the same. Snapshots generated by our animator for the both scenarios are shown in Fig. 3(a) and Fig. 3(b), respectively. Other simulation settings are shown in Fig. 3(c).

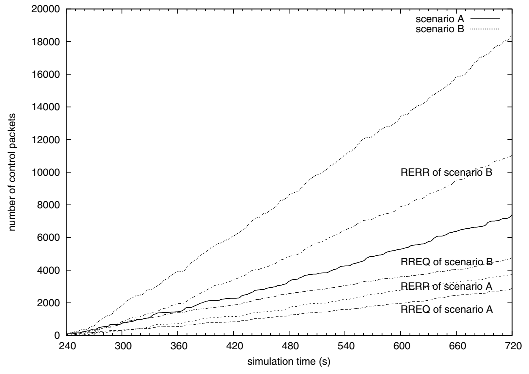
The results in Fig. 4(a) and Fig. 4(b) endorse our estimation. They show the accumulative number of route breaks (we also show the ratio of the number of local recoveries to the number of route breaks called *route refresh ratio*) and the accumulative number of control packets (RREQ and RERR packets) during the simulation time (from time 240 sec. to 720 sec.), respectively. We can clearly see that scenario $B$ has much larger numbers of route breaks and control packets than scenario $A$. Therefore, packet arrival ratio in scenario $B$ (0.665) was lower than that in scenario $A$ (0.864). In this way, we can see the difference in simulation results when we use the realistic pedestrian flows reproduced with MobiREAL.

## 5 Conclusion

In this paper, we have proposed a new mobility model where near real movement of nodes can be easily reproduced. We have also developed a network simulator *MobiREAL* based on the proposed method. Several simulation scenarios are shown on MobiREAL WWW page [1].



(a) Simulation time vs. the accumulative number of route breaks and route refresh ratio.



(b) Simulation time vs. the accumulative number of control packets (RREQ and RERR).

**Figure 4. Experimental results of end-to-end communication example.**

## References

[1] MobiREAL web page. http://www.mobireal.net.

[2] C. Bettstetter. Mobility modeling in wireless networks: Categorization, smooth movement, and border effects. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(3):55–67, July 2001.

[3] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC)*, pages 483–502, 2002.

[4] W. J. Hsu, K. Merchant, H. W. Shu, C. H. Hsu, and A. Helmy. Weighted waypoint mobility model and its impact on ad hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, pages 59–63, 2005.

[5] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proc. of ACM/IEEE Mobicom*, pages 217–229, 2003.

[6] G. F. Riley. The Georgia Tech network simulator. In *Proc. of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pages 5 – 12, 2003.