A Study on Mobility Models in Mobile Wireless Network Simulation

January 2009

Kumiko MAEDA

A Study on Mobility Models in Mobile Wireless Network Simulation

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2009

Kumiko MAEDA

List of Publications

Journal Papers

1. Kumiko Maeda, Kazuki Konishi, Kazuki Sato, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "Development Environment for Mobile Ad-hoc Network Systems Using Network Simulator MobiREAL", *IPSJ Journal*, Vol.47, No.2, pp. 405-414, 2006 (In Japanese).

2. Kumiko Maeda, Takaaki Umedu, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "Development Environment for Mobile Ad-hoc Network Systems Using Network Simulator MobiREAL", *IPSJ Journal*, Vol.48, No.7, pp.2238-2245, 2007 (Recommended Paper) (In Japanese).

3. Kumiko Maeda, Masatoshi Nakamura, Takaaki Umedu, Hirozumi Yamaguchi and Teruo Higashino : "Representing Temporal Density Variation in Urban Pedestrian Flows", *IPSJ Journal*, Vol.49, No.10, pp.3622-3630, 2008 (Recommended Paper) (In Japanese).

4. Kumiko Maeda, Akira Uchiyama, Takaaki Umedu, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "Urban Pedestrian Mobility for Mobile Wireless Network Simulation", *Ad Hoc Networks, Elsevier*, Vol.7, No.1, pp.153-170, 2009 (to appear).

Conferences Papers

1. Kazuki Konishi, Kumiko Maeda, Kazuki Sato, Akiko Yamasaki, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "MobiREAL Simulator - Evaluating MANET Applications in Real Environments", *Proceedings of the 13th Annual Meeting of the IEEE International Symposium*

on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS2005), pp. 499-502, Atlanta, Georgia, USA, Sep. 2005.

2. Kumiko Maeda, Kazuki Sato, Kazuki Konishi, Akiko Yamasaki, Akira Uchiyama, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "Getting Urban Pedestrian Flow from Simple Observation: Realistic Mobility Generation in Wireless Network Simulation", *Proceedings of the 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM2005)*, pp. 151-158, Montreal, Quebec, Canada, Oct. 2005.

3. Kumiko Maeda, Keisuke Nakata, Takaaki Umedu, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "Hybrid Testbed Enabling Runtime Operations for Wireless Applications", *Proceedings of the 22nd ACM/ IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008)*, pp. 135-143, Rome, Italy, June 2008.

Other Related Papers

1. Kumiko Maeda, Takaaki Umedu, Hirozumi Yamaguchi, Keiichi Yasumoto and Teruo Higashino : "MobiREAL : Scenario Generation and Toolset for MANET Simulation with Realistic Node Mobility", *Proceedings of the 7th International Conference on Mobile Data Management (MDM2006)*, CD-ROM, Nara, Japan, May 2006.

2. Akira Uchiyama, Kumiko Maeda, Takaaki Umedu, Hirozumi Yamaguchi and Teruo Higashino : "Performance Evaluation of Mobile Wireless Communication and Services with Modeling of Real Environment", *International Journal of Ad Hoc and Ubiquitous Computing*, Inderscience Publishers, Vol.2, No.4, pp.239-249, 2007.

3. Akira Uchiyama, Sae Fujii, Kumiko Maeda, Takaaki Umedu, Hirozumi Yamaguchi and Teruo Higashino : "Ad-hoc Localization in Urban District", *Proceedings of the 26th Annual IEEE Conference on Computer Communications (INFOCOM2007)*, pp. 2306-2310, Anchorage, Alaska, USA, May 2007.

Abstract

Network simulations play an important role for developing new architecture since it can offer low-cost environment. Since the constituents of wireless networks, especially of Mobile Ad-hoc Networks (MANETs), are not stationary, mobility models greatly affect the performance of wireless network systems. However, there is a tradeoff between the reality of mobility models and the cost for their complexity.

Therefore, I propose Urban Pedestrian Flows (UPF) mobility model. This model targets the reproduction of the walking behavior of pedestrians in city sections, shopping malls and so on using simple observations. In the modeling of real city sections, we only need the average densities of pedestrians on certain streets, which can be obtained by fixed point observations, while many of the existing realistic models require Origin-Destination flows. Given the average densities of pedestrians on certain streets, and a set of walking paths pedestrians are likely to follow, the method estimates flows of pedestrians. The maximum error between the observed density and the derived density is minimized using linear programming techniques to reproduce realistic movement of pedestrians. In the experiment, pedestrians in a 500 m $\times$ 500 m area in front of a large train station in Osaka were modeled and the maximum error between the observed densities and the derived densities was only 8.68 %. Moreover, I have conducted some experiments to clarify the characteristics of the UPF mobility compared with the random-based mobility, and have given two case-studies to investigate the influence of the UPF mobility on the performance of wireless networks.

I also propose some useful methods for mobile wireless network simulation, which are integrated into our wireless network simulator called MobiREAL. One of the main features of the MobiREAL simulator is the mobility description model called the *Condition Probability Event (CPE)* model. A walking person's

relational behavior toward the surroundings can be described by this model, e.g., adjusting walking speeds and directions to avoid collision with neighbors and obstacles and stopping at a traffic signal. Also, the CPE model takes inputs and outputs of network applications into its description space, thus we can describe reactions to the results of the network applications, e.g., a mobile node's detour in response to traffic jam information received through networks. The role of the CPE model is the extension of the basic mobility model like UPF and random way-point. This model allows simulator users to incorporate a microscopic behavior model into a given macroscopic model to enhance the reality and to change a part of mobility depending on the situation without complex coding. I show an evaluation example of an information diffusion protocol with MANET in which the movement paths of some nodes can be dynamically changed depending on the network application status according to given CPE description.

Real-time network simulation is often utilized for evaluating real protocols and applications, and human's perception of the applications. In this research, I propose a real-time simulation technique by prioritizing some simulation events to satisfy their real-timeliness. I focus on the difference of the time-scale between the packet simulation events and the mobility simulation events, and let the MobiREAL simulator process the packet simulation events prior to the mobility simulation events. Through the experiments, I show that the proposed method can reduce the average delay of the prioritized events by up to 25% while keeping the location error which is caused by the delay of mobility simulation events small enough.

The real-time simulation function of the MobiREAL simulator also enables the run-time manual operation of node mobility with the visualized simulation status. Debugs and tests of new protocols and applications still depend on developers' intuition, especially in the early stage of the development. This function allows simulator users to construct simulation scenarios intuitively and flexibly and thus enhances the efficiency of debugging and testing in the development. I show an example that the run-time operation function is used for the evaluation of the DSR routing protocol.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless networks are necessary in the current society and have been improved in various aspects: smaller terminals, longer battery lifetime, faster transmission and so on. Network simulations play an important role for developing new architecture since it can offer low-cost environment for the design, analyze, validate and evaluate new protocols and applications without real terminals and networks. There are many researches considering the fidelity of the network simulation to reduce a gap between the simulation and the real world. Some of them are for the mobility models in the wireless network simulations. Since the constituents of wireless networks, especially of Mobile Ad-hoc Networks (MANETs), are not stationary, mobility models greatly affect the performance of wireless network systems [1–5].

In this paper, we introduce a mobility model called Urban Pedestrian Flows for simulation of wireless networks in city section environment, and a wireless network simulator called MobiREAL which incorporates some useful functions for the mobile wireless network simulations: a relational mobility description model called Condition Probability Event and a run-time mobility operation function under the real-time simulation.

The reality of mobility models is an important factor for the evaluation of the performance of wireless network protocols and applications. However, there is a tradeoff between the reality of mobility models and the cost for their complexity. In particular, modeling movement of real nodes (e.g., pedestrians) with high fidelity for the evaluation of town-wide deployed networks requires detailed observation or survey of those people moving from place to place along streets. Obviously such observation cannot be easily carried out due to cost, privacy and other reasons.

Therefore, we propose the Urban Pedestrian Flows (UPF) mobility model. This model is aimed at reproducing the walking behavior of pedestrians in city sections, shopping malls and so on using simple observations. Because most of the existing realistic mobility models target cellular networks or hotspots of wireless LANs in which networks are not so dynamic compared to the MANETs and mobile sensor networks, walking paths are abstracted in those models. However, my proposed UPF model reproduces pedestrians' walking behavior along streets in city sections. This model considers reality of mobility in two aspects; pedestrians walk along the one of the given walking paths, and under the constraint, the flow rate of each path is derived to well match given density on each road. Thus, the UPF model is suitable for the evaluation of MANETs and sensor networks. In the modeling of real city sections, we only need the average densities of pedestrians on certain streets, which can be obtained by fixed point observations, while many of the existing realistic models require Origin-Destination flows. Given the average densities of pedestrians on certain streets, and a set of walking paths pedestrians are likely to follow, the method estimates flows of pedestrians. The maximum error between the observed density and the derived density is minimized using linear programming techniques to reproduce realistic movement of pedestrians. In the experiment, two persons have measured the average densities of pedestrians on 33 streets in a 500 m $\times$ 500 m area in front of a large train station in Osaka for about a half hour. The maximum error between the observed densities and the derived densities was only 8.68 %. To clarify the character of UPF mobility, we have compared UPF mobility and random-based mobility by measuring some metrics that characterize mobility and connectivity of network topology graph presented in Ref. [2]. The performance of the DSR routing protocol with both mobility models is also shown. Moreover, we show the influence of the characteristics of UPF mobility on the performance of wireless networks through two case-studies, real-time data streaming from a stationary base station to a mobile client via MANET, and a mobility management protocol for wireless mesh networks using the mobility prediction. Through these experiments, we have investigated the influence of the node mobility on the performance of MANET protocols and have shown the

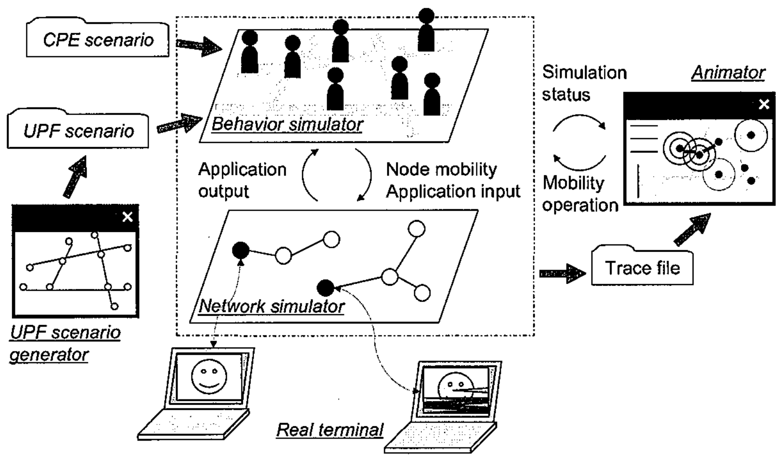


Figure 1.1: Overview of the MobiREAL simulator

application and the usefulness of the UPF model.

In addition to the above mobility model, we propose some useful methods for mobile wireless network simulation, which are integrated into the wireless network simulator MobiREAL. Fig. 1.1 shows the overview of the MobiREAL simulator. MobiREAL supports the UPF mobility model and offers a GUI tool to help generation of UPF scenarios. MobiREAL simulator supports both real-time simulation and genuine simulation so that real application codes on real mobile terminals can be connected to a simulated network of MobiREAL.

One of the main features of the MobiREAL simulator is the mobility description model called the *Condition Probability Event (CPE)* model. A walking person's relational behavior toward the surroundings can be described by this model, e.g., adjusting walking speeds and directions to avoid collision with neighbors and obstacles, and stopping at a traffic signal. Also, the CPE model takes inputs and outputs of network applications into its description space, which allows describing reactions to the results of the network applications, e.g., a mobile node's detour in response to traffic jam information received through networks. The mobility and the network status affect each other in this situation because the change of movement causes the change of network topology, thus the different performance may be obtained. This model allows simulator users to integrate various behavior models in the network simulation without complex coding. The role of the CPE model is the extension of the basic mobility model like UPF and random way-point. Users can incorporate a microscopic behavior model into a given macroscopic model to enhance the reality and can change a part of mobility depending on the situation to create an objective simulation scenario. As long as we know, this is the first approach that focuses on such interdependency between mobility and network.

The CPE model requires the interaction between the mobility simulation module and the network application simulation module. we implemented the CPE simulation module including the mobility simulation module as a separete program (behavior simulator) from a network simulation module (network simulator), and prepared the interface to the network simulator with the socket communication considering the compatibility with the existing simulators. The MobiREAL simulator is developed by extending the network simulator GT-NetS [6] to cooperate with the behavior simulator. The behavior simulator and some GUI tools of MobiREAL can be utilized for other network simulators with slight extension, or without extension in case of using a trace-based mobility model supported by most network simulators. We show an evaluation example of an information diffusion protocol with MANET in which the movement paths of some nodes can be dynamically changed depending on the network application status according to given CPE description.

Real-time network simulation is often utilized for evaluating real protocols and applications, and human's perception of the applications. Some simulators and emulators are developed for such a purpose. In this research, we propose a real-time simulation technique by prioritizing some simulation events to satisfy their real-timeliness and implemented it into the MobiREAL simulator. We focus on the difference of the time-scale between the packet simulation events and the mobility simulation events, and let the MobiREAL simulator process the packet simulation events prior to the mobility simulation events. Through the experiments, we show that the proposed method can reduce the average delay of the prioritized events by up to 25% while keeping the location error which is cased by the delay of mobility simulation events small enough.

The benefit brought by the real-time simulation is not only the cooperation with real codes but also the run-time manual operation of node mobility with the visualized simulation status. Manual operation function with the visualized simulation status allows simulator users to construct simulation scenario intuitively and flexibly and thus enhances the efficiency of debugging and testing in the development. Debugs and tests of the new protocols and applications still depend on developers' intuition, especially in the early stage of the development. The run-time operation function helps developers to create the scenario, e.g., for the validation of time-critical and state-critical behavior. A typical example is assessment of multi-hop MANET routing strategy; we may want to disconnect established routes by removing a relay node in various situations for testing the reliability of the route repair process. It is hard to predict on which node the route is established if the protocol contains various artifices, e.g., multiple routes and route caches strategies. We often have to try and repeat simulations many times to reach an objective scenario. The run-time operation function of MobiREAL simulator helps such a process, e.g., the visualized simulation status helps to understand the progress of simulations and the manual operation and the immediate reflection of the change cuts-down the cost of the repeating simulations. The animator of MobiREAL can be used to show the real-time progress of simulations. This tool can animate the movement of nodes, network topology, packet propagation and so on, and can also show statistical information like node density and packet error ratio observed in each sub-region. The run-time operation of nodes' movements and the simulation progress can be done through this tool. We show an example that the run-time operation function is used for the evaluation of the DSR routing protocol.

The rest of this paper is organized as follows: Chapter 2 explains the Urban Pedestrian Flows mobility model with related experiments. Chapter 3 shows the Condition Probability Event model with an application example. Chapter 4 explains the design and architecture of MobiREAL simulator; in particular, the cooperation between the behavior simulator and the network simulator for the CPE model and the implementation for the real-time simulation and the run-time operation is explained. The performance of MobiREAL simulator is also shown. Chapter 5 describes the proposed real-time simulation technique. Chapter 6 concludes this paper and outlines further research issues.

Chapter 2

Urban Pedestrian Flows Mobility Model for Wireless Network Simulations

In this chapter, we introduce Urban Pedestrian Flows (UPF) Mobility Model. This model considers the reality of mobility in two aspects, walking paths of pedestrians and the distribution of the density on each road. Pedestrians walk along the one of the specified paths. Under the above constraint, the flow rate of each path is derived to well match specified density using linear programming (LP) techniques. We describe related works in Section 2.1 and details of UPF model in Section 2.2. A modeling example and some metrics of the UPF mobility and the random-based mobility that characterize mobility and connectivity of network topology graph are shown in Section 2.3, and two case studies are shown in Section 2.4.

2.1 Related Work

We often use simple mobility models such as the Random Way Point (RWP) model [7] in free space for simulating MANET protocols. This is because such simple mobility models are commonly available in many simulators and therefore can be a common environment for comparative experiments. Several analytical results have been presented for the RWP model and its variants [8–18]. In [8], Bettstetter presents a variant of RWP where nodes can change their directions smoothly, keeping the same analytical properties as RWP. In [9], Chu and Nikolaidis address that the node density of RWP is non-uniform and that the non-uniformity depends on the speed of nodes. In [10], Rojas et al. show that the Cauchy distributions and the chi-square distributions can model the deployment of locations and residual pause times of waypoints more accurately than uniform distributions and exponential distributions, respectively. In [11], Hyytia et al. derive an expression that represents the nodes' position distribution of RWP in an arbitrary convex domain and propose the RWPB model that forms contrastive distribution with RWP. In [12], Yoon et al. focus on the velocities of nodes in RWP and prove that the harmful effects are observed in the performance evaluation before reaching the steady-state. Yoon et al. also present the "sound mobility model" based on the RWP model in [13]. This model is designed to maintain the average velocities of nodes to exclude the harmful effects. In [17], McGuire derives the stationary distribution of nodes for a general class of mobility models. In [18], Nain et al. analyze the stationary distribution of nodes in the random direction model and describe its usefulness compared with RWP.

There are many methodologies that synthesize mobility models from observed data and geography information to pursue reality [19–24]. In [19], Jardosh et al. introduce obstacles (i.e., buildings) and pathways in a given simulation field. A pathway between any two obstacles can automatically be generated using a Voronoi graph computation algorithm. The research group also provides plug-ins for GloMoSim [25] and ns-2 [26] simulators to utilize the proposed mobility. In [20], Hollick et al. propose a macroscopic mobility model for wireless metropolitan area networks, where a simulation field is divided into multiple zones with different attributes such as workplace, commercial and recreation zones. Also, each mobile node has an attribute of resident, worker, consumer or student. Given trips with destinations for user nodes, an existing urban transportation planning technique is used to estimate the user density in each zone. In [21], Hsu et al. present the WWP (Weighted Way Point) model. The WWP model defines a set of crowded regions such as cafeterias and buildings at a university. Given a distribution of pause times for each region and a transition probability of nodes for each pair of regions, it uses a Markov model to model nodes' movement between these regions. For vehicular ad hoc networks, a technique to reproduce realistic mobility of vehicles in ns-2 is proposed in [22].

Group-based mobility models also capture some aspects of realistic movement. In [23], Musolesi et al. propose a group-based mobility model based on social networks of individual users. Also, several group-based mobility models are proposed in [24].

There is some research analyzing the relationship between network traffic and the behavior of users by tracing user behaviors on wireless networks. In [27], Kotz and Essien analyze user behavior patterns by collecting traces of 2,000 users for 11 weeks at 476 wireless APs distributed in 161 buildings at Dartmouth College. They found interesting characteristics of traffic depending on time and location. The authors of [28–30] have also collected similar traces from wireless network users in a metropolitan area wireless network at the SIGCOMM2001 conference location and in three large corporate buildings, respectively. In [31], Thajchayapong and Peha obtained traces from the IEEE 802.11-based system at the Carnegie Mellon University campus. Contrary to most researchers' expectations, the cell residence times do not form an exponential distribution. In [32], McNett and Voelker analyze the mobility patterns of users of wireless hand-held PDAs in a campus wireless network and synthesize the "campus waypoint model" that serves as a trace-based complement to the RWP model. Also, Kim et al. extract user mobility characteristics from wireless network traces and present a mobility model based on the characteristics in [33]. These research efforts aim at getting knowledge on capacity planning and AP arrangement for building next-generation large-scale mobile network infrastructures. However, they require some wireless network infrastructures and a very large amount of information collection to produce a mobility scenario.

Many realistic mobility models including my proposed methodology have intended to model the environment in the real world. My proposed methodology is closest to the work presented in [19–21], which tries to reproduce the real world's geography and movement of nodes. These methodologies require some observation of user behaviors in a target region, but the simplicity of observation is an important factor because a large amount of human and system resources are required to collect complete information about users' behavior. Therefore, we propose a methodology to reproduce urban pedestrian flows from the density of nodes. Densities of pedestrians can be obtained by fixed-point observation using cameras and therefore it is easier than measuring transition probabilities among hotspots, which is assumed by the methods presented in [20,21]. The UPF model considers the node density to reproduce the nodes' moving flows over streets, while many existing models use the node density at the hotspots (i.e. major points) to reproduce the nodes' transitions among them. The UPF model possibly fails to accurately reproduce the transition probability among the major points, but can reproduce realistic flows on streets in the city section, which are abstracted in those existing models.

2.2 Deriving Urban Pedestrian Flows

To generate a UPF scenario, we give a street map of the simulation field as a graph called a *street graph*, and possible paths of pedestrians on the street graph. We also give average densities of pedestrians on some streets. These densities can be simply obtained by fixed-point observations and so on. Then for each path, we calculate the number of pedestrians that come into the path per unit of time (this number is called *flow rate*) using a linear programming (LP) technique, where the maximum error between the density obtained from the observation and that derived from the LP solution is minimized. Using the derived flow rates, the UPF scenario generation tool generates a UPF scenario that can be used in the MobiREAL simulator. This will be described later.

2.2.1 User Inputs

Simulation Field

A simulation field may contain polygons that represent obstacles such as buildings and an undirected street graph $G = (V, E)$ where V and E represent geographic points and streets, respectively (Fig. 2.1). The geographic points are set at intersections and entrances to buildings, stations, terminals, underpasses, shopping centers and so on. For each edge $e_{ij} \in E$, the width W_{ij} of the street is also given.

Pedestrians' Paths

In urban areas, many types of pedestrians are walking around for different purposes. Commuters usually get off trains and go straight to their transfer points or offices. Shoppers visit their favorite shops and remain for a while. We

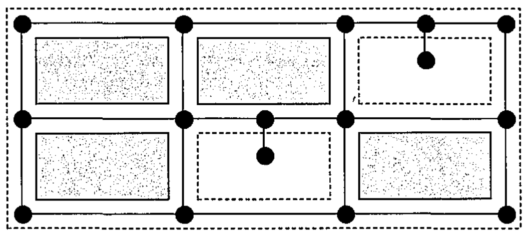


Figure 2.1: Simulation field and street graph.

estimate the possible walking paths of pedestrians and represent them as a set R of acyclic paths on street graph G. Each path starts and ends at vertices that can be starting points and destinations like stations and buildings. We note that if a path contains a loop, we may regard this loop as another path and separate it from the original path. Then we can avoid loops even though the given paths contain them.

It may be difficult to enumerate all of the possible paths. Therefore, we may use the following simple enumeration. In general, pedestrians are considered to walk along the shortest paths. So, we specify some points where many people appear or disappear (e.g. stations and shops) and compute the shortest path for every pair of these points. For each path computation, we can choose one of several options case by case. For example, considering the fact that people tend to walk along major streets, we put some weights that prioritize these streets in the shortest path construction algorithm.

Observed Density of Pedestrians

Next, we have to determine a flow rate of each path, that is, the number of pedestrians that come into the path per unit of time. We give the densities on some streets measured in the real world and derive flow rates based on these values. We let $E_M (\subseteq E)$ denote the set of edges on which densities of pedestrians are observed. For each edge $e_{ij} \in E_M$, we let D_{ij} denote the observed density on the edge.

In general, for the node density d on an edge, the *pedestrian rate g on the edge* (i.e., the number of pedestrians going into the edge per second), the speed v of pedestrians on the edge and the width w of the edge, the following equation holds.

$$d_{(person/m^2)} = \frac{g_{(person/s)}}{w_{(m)} \cdot v_{(m/s)}} \tag{2.1}$$

It is known that the speed of a pedestrian follows the negative increase of density. Therefore,

$$v = -k \cdot d + v_0 \tag{2.2}$$

where k and v_0 are positive constants. Here, v_0 is the average speed of the pedestrian in a sparse area. Also on any (very) crowded street, a particular density (say D_{max}) was observed when the speeds of pedestrians are very close to 0. Therefore, from (2.2), we obtain

$$-k \cdot D_{max} + v_0 = 0 \tag{2.3}$$

Finally, from equations (2.1), (2.2) and (2.3), we obtain the following pedestrian rate calculation function G from the density d and width w of the edge.

$$g = G(d, w) = v_0 \cdot w \cdot d \cdot \left(1 - \frac{d}{D_{max}}\right) \tag{2.4}$$

We should select streets to observe by considering the characteristics of the field geography and the desired preciseness of the reproduction. The solution will become more precise if we obtain more information about densities. We evaluate the relationship between the number of streets and the preciseness of the result in Section 2.3.2.

2.2.2 Determining Flow Rates

Next, we describe the algorithm to derive flow rates from the observed pedestrian density. Given an undirected graph G, the width W_{ij} of each edge $e_{ij} \in E$, a set R of paths, and the observed density D_{ij} on each edge $e_{ij} \in E_M$, the algorithm derives the *flow rate* of each path in R, minimizing the maximum error between the derived and observed densities on the edges.

To formulate this problem as an LP problem, we introduce a variable f_k representing the flow rate of path $r_k \in R$ and a variable g_{ij} representing the pedestrian rate on edge $e_{ij} \in E$. We also introduce a variable δ, which means the maximum error of the derived pedestrian rate from the corresponding pedestrian rate given by equation (2.4).

For each edge e_{ij}, its pedestrian rate is the sum of flow rates on e_{ij}. Therefore, the following equation holds.

$$\forall e_{ij} \in E; \;\; g_{ij} = \sum_{r_k \in R \wedge e_{ij} \in r_k} f_k \tag{2.5}$$

The following inequality limits the errors of derived pedestrian rates by δ, where function G is given by equation (2.4).

$$\forall e_{ij} \in E_M; -\delta \leq \frac{G(D_{ij}, W_{ij}) - g_{ij}}{G(D_{ij}, W_{ij})} \leq \delta \tag{2.6}$$

Also pedestrian rate g_{ij} has an upper limit derived from the nature of function G. From the definition of G in (2.4),

$$\begin{aligned} g_{ij} &= G(d_{ij}, W_{ij}) \\ &= v_0 \cdot W_{ij} \cdot d_{ij} \cdot \left(1 - \frac{d_{ij}}{D_{max}}\right) \\ &= -\frac{v_0 \cdot W_{ij}}{D_{max}} d_{ij}^2 + v_0 \cdot W_{ij} \cdot d_{ij} \\ &= -\frac{v_0 \cdot W_{ij}}{D_{max}} (d_{ij} - \frac{D_{max}}{2})^2 + \frac{v_0 \cdot W_{ij} \cdot D_{max}}{4} \end{aligned}$$

is obtained. Knowing that v_0, W_{ij} and D_{max} are all positive constants, the following inequality is obtained.

$$g_{ij} \leq \frac{v_0 \cdot W_{ij} \cdot D_{max}}{4} \tag{2.7}$$

Our objective is to minimize the maximum error δ. Under constraints (2.5), (2.6) and (2.7), the objective function is defined as follows.

$$\min \delta \tag{2.8}$$

By solving this LP problem, we obtain the flow rate f_k for each path $r_k \in R$ minimizing the maximum error δ of the derived pedestrian rates. Using the derived flow rates, a UPF mobility scenario that can be used in MobiREAL is generated automatically by the UPF scenario generation tool. The details of this tool are given in Section 4.2.2. In the scenario, we generate node instances following the derived flow rate f_k, and the nodes walk along the path r_k. A path with flow rate 0 means that there is no pedestrian flow along the path.

2.3 Experiments

We have carried out several experiments. My objective is three-fold. First, we would like to confirm that my UPF methodology and the related MobiREAL toolset can model the pedestrians' flows in the real world with high fidelity, without a great deal of manpower. For this purpose, we conducted fixed-point observation in downtown Osaka and derived the pedestrians' flows. The result of this experiment given in Section 2.3.1 has shown that my scheme could reproduce a position distribution very similar to the observed ones. Second, we would like to evaluate the impact of the amount of given density information in the derivation of the UPF scenario in Section 2.3.2. Section 2.3.3 clarifies characteristics of UPF model by comparing some metrics of UPF mobility and random-based mobility.

2.3.1 Fidelity of Reproduced Pedestrian Flows

We have measured the node density on each street in downtown Osaka. Using the obtained data, we have determined pedestrian flows based on my UPF technique described in Section 2.2 to see the similarity of the derived and observed density of pedestrians.

Two students observed the node density on the streets beginning at 14:00 on Sunday. At each observation point, we just took a photo with a digital camera, so the observation period itself was an instant. Next, we measured the width and length of the street captured in the photo. Finally, we calculated the density of pedestrians using those values and the number of pedestrians in the photo. The obtained data is shown in Table 2.1. The map of downtown Osaka with the corresponding street graph is shown in Fig. 2.2. The size of the field is 500 m $\times$ 500 m and the graph includes 35 vertices and 44 edges. It took two students only about a half hour to obtain the average densities on the 43 streets.

As candidates for nodes' source or sink points (destination points), we selected 21 points located near shops, major intersections with subway entrances and the border of the field. We calculated the shortest path for each pair of the 21 destination points, and thus 210 routes were found as a total. We assumed that $v_0 = 1.39\ m/s$ and $D_{max} = 1.4\ person/m^2$. Then we constructed 130 linear constraints that included 253 parameters (variables). lp_solve [34] was used to solve the given LP problem and the solving time was less than one second.

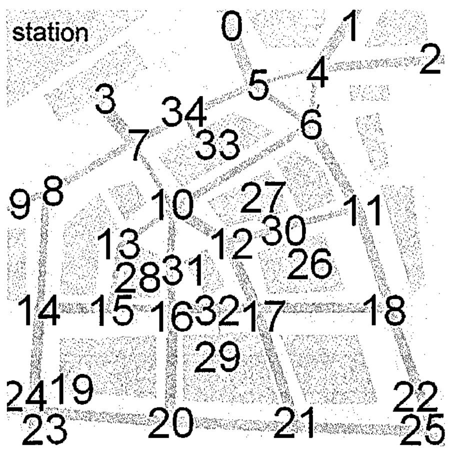


Figure 2.2: Downtown Osaka.

We show the result in Table 2.2. When we obtained the solution of the LP problem, the maximum error, which was minimized in the LP problem, was 9.09 % (0.102 person/s) and the mean error was 8.74 % (0.030 person/s). We would like to note that since there is no absolute benchmark to compare with, there is no way to thoroughly show the fidelity of this result. However, in the general consensus we can say that this error is in the acceptable range.

These simulation settings and the obtained UPF mobility were also used in the experiments described in the following subsection.

2.3.2 Density Specified Ratio and Reproduction Quality of Derived Flows

In this experiment, we evaluated the impact of the amount of given density information on the accuracy of UPF mobility. We observed the node densities on the 43 streets as mentioned in Section 2.3.1. In this section, we used the part of the observed 43 density information (from 22 up to 39 chosen randomly from the 43 density information) as an input of LP and evaluated the error of the density on the derived UPF flows. Table 2.3 shows the average/maximum error between the observed pedestrian rate and the derived one ($|observed - derived|$) and its ratio ($\frac{|observed-derived|}{observed} \cdot 100$). Here, "specified" is the error only on the streets where density was given, but "all" is the error on all streets. For the

Table 2.1: Measured node density on streets in downtown Osaka.

Edge	Width	Density	Edge	Width	Density
9-8	7	0.033	7-10	6	0.065
8-7	8	0.059	10-31	8	0.046
7-34	8	0.041	31-16	8	0.045
34-5	8	0.041	16-20	12	0.008
5-4	4	0.025	10-12	12	0.015
4-2	10	0.013	18-22	12	0.026
15-13	6	0.005	12-17	12	0.012
13-10	6	0.005	17-21	12	0.012
10-6	8	0.030	11-30	8	0.040
14-15	12	0.008	30-12	8	0.040
15-16	8	0.006	0-5	14	0.034
16-32	12	0.012	5-6	7	0.050
32-17	12	0.012	6-11	12	0.060
17-18	12	0.013	11-18	12	0.021
24-19	12	0.032	1-4	12	0.002
19-20	12	0.035	4-6	5	0.030
20-21	12	0.023	30-26	3	0.013
21-22	12	0.027	30-27	3	0.066
8-14	8	0.045	28-31	3	0.025
14-19	12	0.026	32-29	3	0.012
19-23	8	0.058	34-33	3	0.053
3-7	14	0.060			

width : m, density : person/m^2

Table 2.2: Derived flow rates.

Path	Flow Rate	Path	Flow Rate	Path	Flow Rate
g11_4	0.1117	g6_18	0.6987	g2_20	0.0205
g24_11	0.0396	g6_21	0.0103	g2_28	0.0958
g24_16	0.0545	g11_21	0.1590	g2_33	0.0481
g24_25	0.3574	g27_11	0.0963	g0_1	0.0048
g24_33	0.0063	g27_16	0.0058	g0_3	0.2536
g29_11	0.0074	g27_33	0.1407	g0_6	0.4292
g29_18	0.0380	g4_14	0.0729	g1_2	0.0364
g29_33	0.0088	g9_11	0.0463	g23_20	0.1694
g18_14	0.0290	g9_18	0.0346	g23_24	0.0286
g18_20	0.1009	g9_23	0.0529	g3_9	0.2105
g6_11	0.0780	g25_11	0.3497	g0_33	0.0306
g6_14	0.0081	g25_16	0.0479	g3_16	0.2433
g6_16	0.1728	g25_26	0.0502	g3_23	0.3170

Flow Rate : person/s, flow rates of all the other routes were zero.

Table 2.3: Density specified ratio and error of pedestrian rate.

# of specified streets	43	39	34	30	26	22	rwalk
# of constraints	130	122	112	104	96	88	–
specified average	0.0304 (8.74)	0.0273 (7.81)	0.0204 (5.91)	0.0139 (3.98)	0.0112 (3.25)	0.0069 (2.04)	– –
specified maximun	0.102 (9.09)	0.102 (9.09)	0.102 (9.09)	0.101 (9.00)	0.0984 (8.82)	0.0938 (8.82)	– –
all average	0.0304 (8.74)	0.0445 (13.7)	0.0645 (21.7)	0.0861 (31.2)	0.111 (43.7)	0.144 (59.5)	0.270 (253)
all maximum	0.102 (9.09)	0.449 (135)	0.626 (265)	0.738 (408)	0.801 (622)	0.933 (901)	0.825 (2170)

person/s (%)

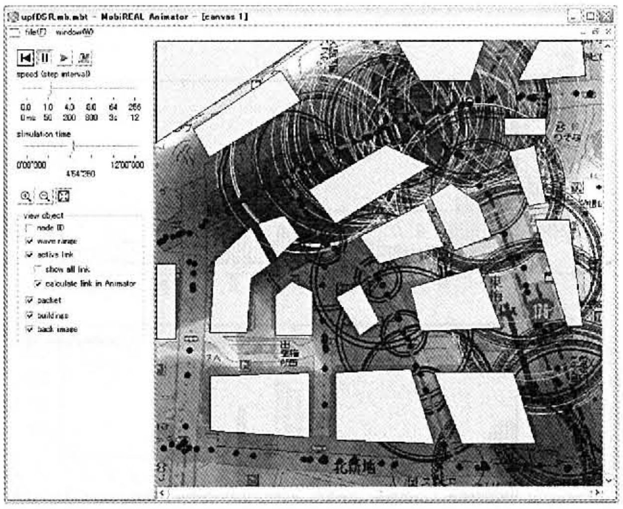
comparison, we also simulated a modified version of the Random Walk mobility called *Random Walk with Obstacles (rwalk/ob)* and counted the pedestrian rate of the model. In rwalk/ob, each node moves along the given street graph. At each corner, the node selects the neighboring corner at random and moves toward it. We used the same street graph in both UPF and rwalk/ob. We also show the error between the pedestrian rate of the rwalk/ob and the observed rate.

The number of LP constraints decreases as the amount of given density information (the number of specified streets) decreases, but the number of LP variables is the same for all cases. The error on the specified streets decreases as the number of specified streets decreases because the constraints are relaxed, but the error on the unspecified streets increases. The error on the unspecified streets varies from 0 to 900. The result shows that losing little density information tremendously increases errors at specific points. With more detailed analysis, we found that the observed density that extremely differs from the neighbors tends to cause large error if such density information is not given to the LP problem. Consequently, we should observe as many streets as possible, being especially careful about the streets with high or low node densities. As future work, we will study a method to fill in the unknown densities from the known densities by using empirical knowledge.

2.3.3 Characteristics of UPF and Influence on Network Performance

In this subsection, by comparing UPF mobility and random-based mobility, we investigate the influence of UPF on the performance evaluation of MANET. We used DSR protocol in the simulation. We measured the metrics that characterize mobility and topology graph connectivity presented in Ref. [2] and the performance of the DSR protocol. Mobility affects topology graph and the performance of the protocol is influenced by topology graph.

We have measured the performance of two applications on the UPF mobility and a modified version of RWP mobility called RWP with obstacles (denoted as RWP/ob). In RWP/ob, each node moves along the same road structure graph used in UPF scenario. At each intersection, the node randomly decides the next direction. The total number of nodes of RWP/ob model is controlled so that it



[ ] node ID
[✓] wave range
[✓] active link
[ ] show all link
[✓] calculate link in Animator
[✓] packet
[✓] buildings
[✓] back image

Figure 2.3: Snapshot of DSR scenario.

has almost the same number of nodes as that of the UPF scenario.

In DSR, the simulation time was 720 seconds. The communication was done by UDP from time 240 sec. to time 720 sec. 10 kbps CBR traffic was generated interactively between the two application users. We used IEEE 802.11 DCF with RTS/CTS as the MAC protocol. The radio range was set to 100 meters. The speeds of nodes were randomly selected between 1.1m/s and 1.7m/s.

At first, we have measured the mobility characteristics metrics; (i) degree of spatial dependence (difference of velocities between two nodes whose distance is less than 50m), (ii) degree of temporal dependence (difference of velocities between two time slots of a node), and (iii) relative speed of two nodes whose distance is less than 50m. The results are shown in Fig. 2.4, 2.5 and 2.6. In order to know the details and definition of these metrics, you can see Ref. [2].

In spatial dependence and relative speed, the results of UPF and RWP/ob were very similar to each other. But in temporal dependence there was big difference. In UPF, each node does not change his/her direction at intersections because each node goes to his/her destination through the shortest path. Consequently, its dependence becomes very high though it is not in RWP/ob.

Secondly, we have measured graph connectivity metrics; (i) link changes (the number of link creations between two nodes) and (ii) link duration (the longest

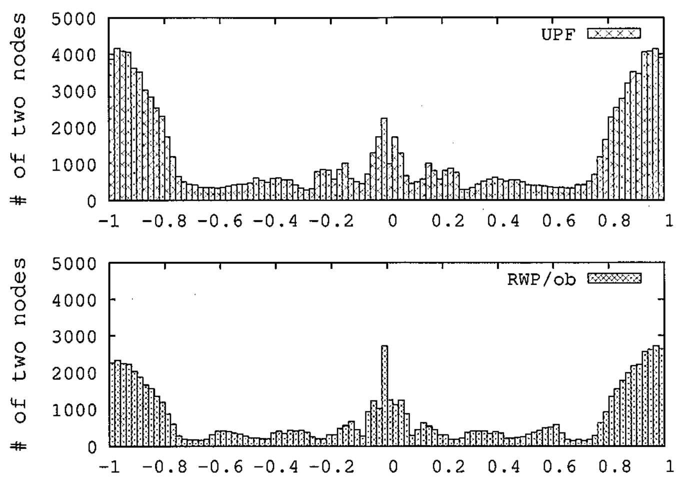


Figure 2.4: Degree of spatial dependence.

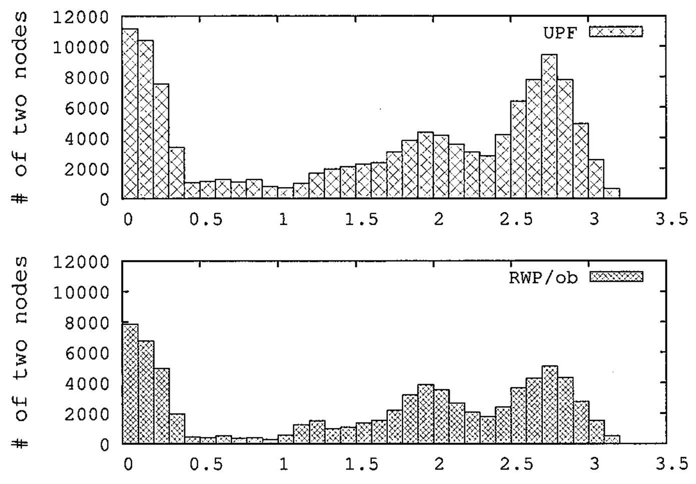


Figure 2.5: Degree of temporal dependence.

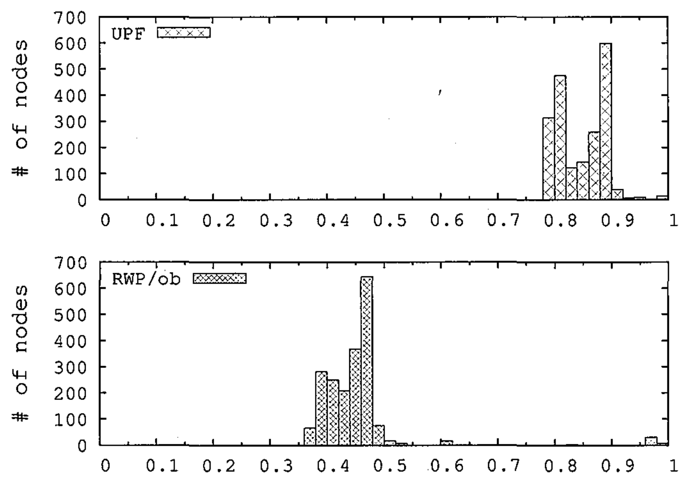
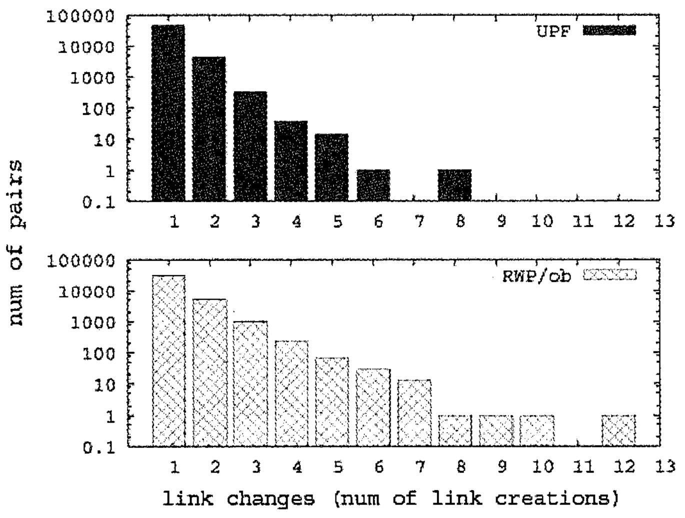


Figure 2.6: Relative speed of two nodes.

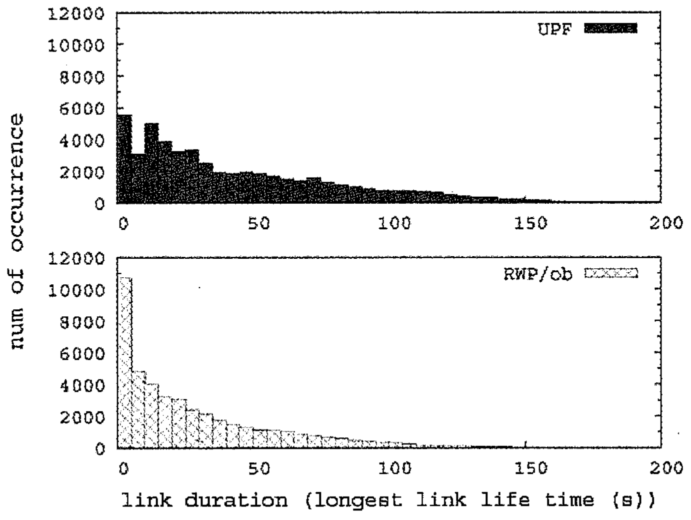
time interval during which two nodes are in the transmission range of each other). The distributions of these metrics are shown in Fig. 2.7. RWP/ob model has several cases with larger link changes compared with my UPF mobility. As mentioned above, probability that nodes change their direction is lower in UPF. Thus, in UPF, the cycle which a neighboring node replaces is longer. So the disconnection of the link does not occur frequently. This observation is endorsed by the link duration result (Fig. 2.7(b)), where UPF has longer durations clearly.

At last, we have measured several metrics that show the performance of DSR. The results are shown in Fig. 2.8, 2.9, 2.10 and 2.11. As mentioned above, in the UPF scenario, a DSR route is stabler than RWP/ob. This observation is endorsed by Fig. 2.8 and Fig. 2.9. The packet arrival ratio is higher and the number of control packets is smaller in UPF than in RWP/ob. Finally in Fig. 2.11, RWP/ob has similar hops in many cases, since the density on any street was almost the same among all streets. On the other hand, UPF has different density among streets, and therefore they are distributed widely.

Through this experiment, we found that the performance of network system differs in random based mobility and UPF mobility even if we use the same road structure. and that there is an influence on the performance caused by uneven density which is not reproduced by the random base mobility.

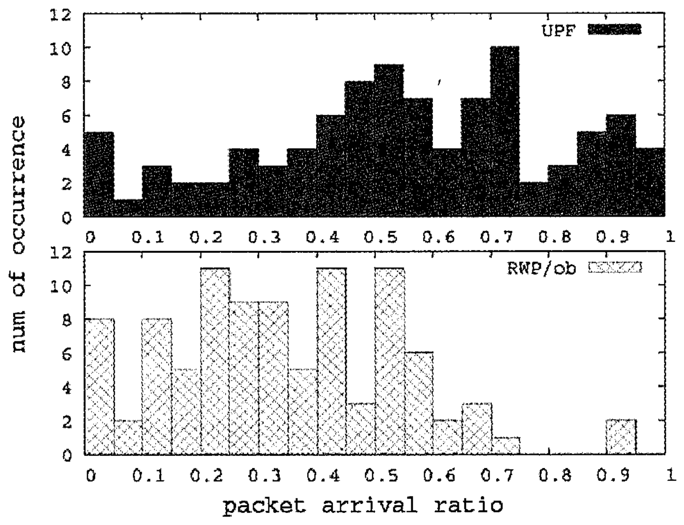


(a) Link changes



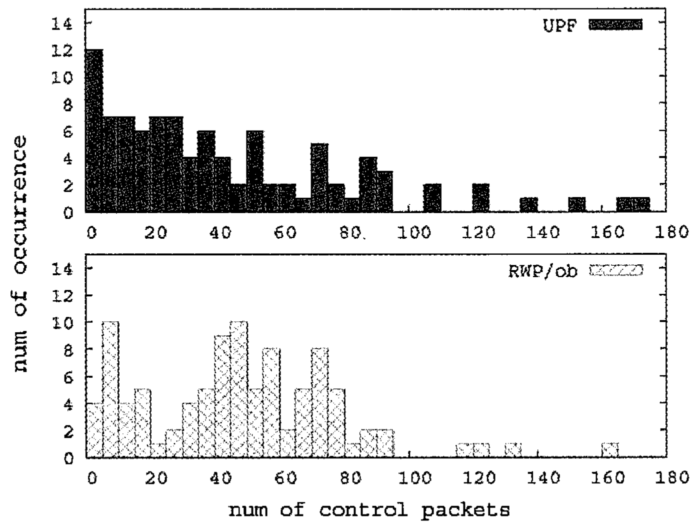
(b) Link duration

Figure 2.7: Graph connectivity metrics.



(captured during every 5s and the averages per second are shown)

Figure 2.8: Distribution of packet arrival ratio.



(captured during every 5s)

Figure 2.9: Distribution of number of DSR control packets.

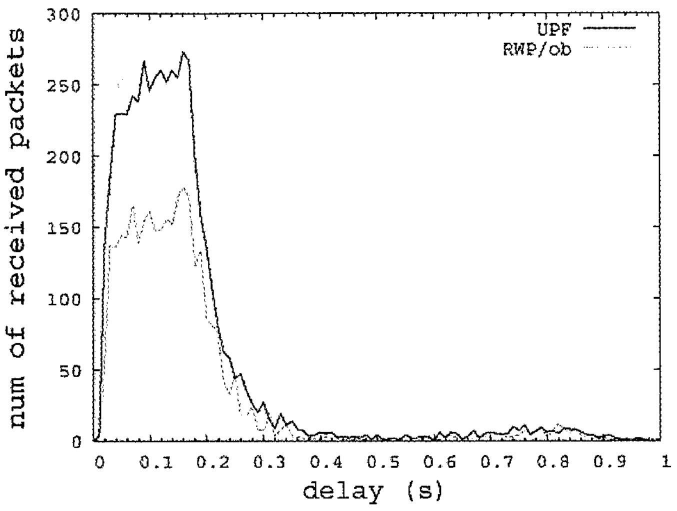
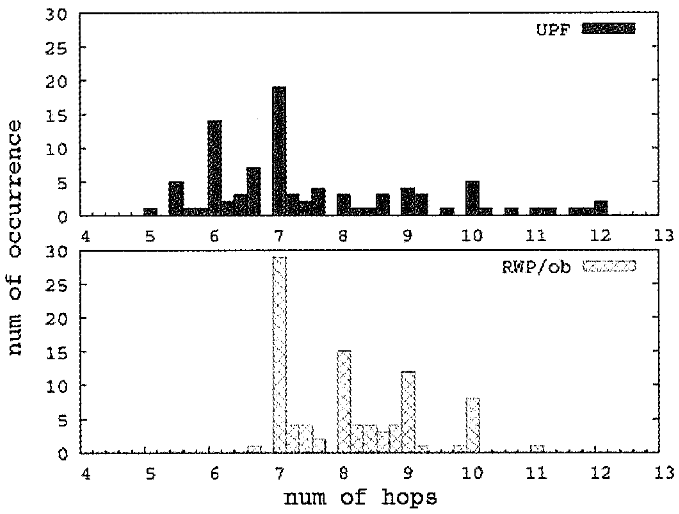


Figure 2.10: Distribution of packet delay.



(captured during every 5s and the averages per second are shown)

Figure 2.11: Distribution of DSR hops.

2.4 Case Study

We would like to show that the non-uniform distribution of the node movement and position affects performance of mobile wireless network systems. For example, in the real world, the location of wireless base stations considerably affects the quality of multi-hop wireless network services while it will have little impact with uniform movement and position distribution of nodes. This fact encourages me to assert the effectiveness of my UPF model presented in this paper. Thus, to prove the fact, we conducted performance evaluation of the following application and protocol: (i) real-time data streaming from a stationary base station to a mobile client via MANET, and (ii) a mobility management protocol for wireless mesh networks using the mobility prediction as in [35, 36].

2.4.1 Data Streaming on MANETs

In this experiment, we carried out simulation of real-time data streaming with the assistance of MANETs. In the simulation scenario, real-time data such as movies are delivered by multi-hop relay from a short-range base station such as an Internet gateway to mobile clients. A route is established by the DSR protocol from the base station to each client. Through this scenario, we examined my expectation that different placements of base stations cause considerable performance distinction if realistic (thus non-uniform) position distribution is assumed, but little difference under uniform ones. If this is true, this fact emphasizes the necessity of realistic mobility.

Figure 2.12 (a) shows the simulation field. In the simulation, each client receives a 56 kbps movie for 180 seconds. We chose 120 clients at random at every 180-second interval. We compared my UPF mobility scenario modeled in Section 2.3.1 and Random Walk with Obstacles (rwalk/ob). We used the same street graph in both UPF and rwalk/ob. We used IEEE802.11 DCF with the RTS/CTS mechanism as the MAC protocol. The radio range was set to 100 meters. In the simulation, we used a simple radio attenuation considering line-of-sight propagation only. The obstacles were placed as shown by the grayed polygons in Fig. 2.12 (a).

We chose three candidate points for the location of the base station where node densities are quite different from each other in the UPF model. To measure the node density in the field, we may exploit the capability of the Animator,

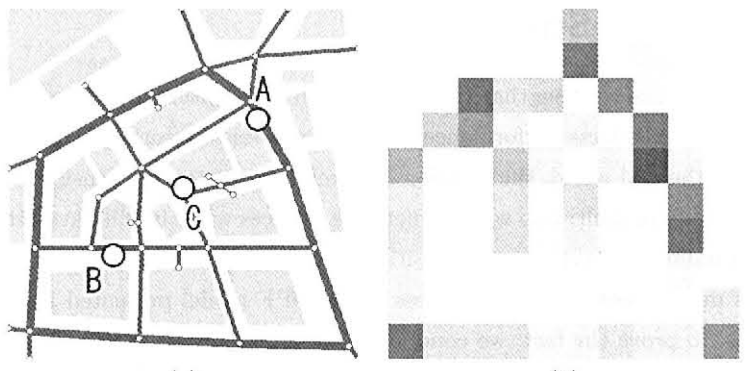
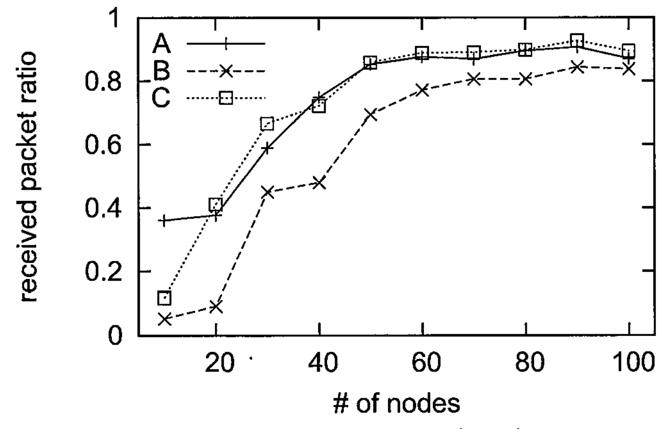


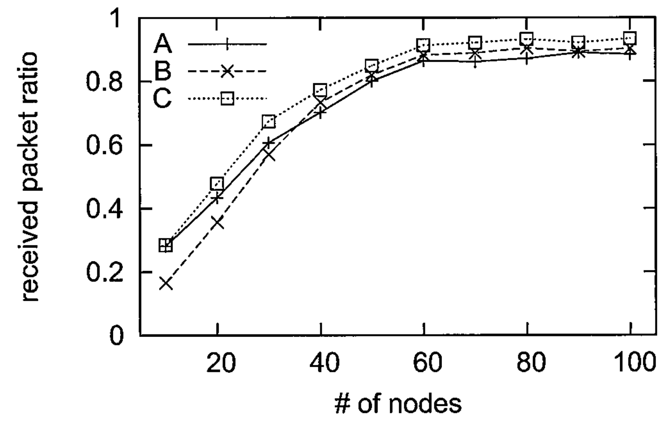
Figure 2.12: (a) Simulation field and candidate points for location of base station. (b) Graphical presentation of node density (snapshot from the MobiREAL animator).

which can show the node density in each square section as shown in Fig. 2.12 (b). In this figure, the depth of gray in each section represents the degree of node concentration (*i.e.* deeper gray means higher density). Then we selected the three points A, B and C in Fig. 2.12 (a) as the candidate positions for a base station placement. A, B and C correspond to high-, low- and middle-density sections, respectively. Then we measured the packet arrival ratio and the average path length to see the performance difference.

Figure 2.13 shows packet arrival ratio at the destination, and Fig. 2.14 shows the average hop length to the destination. As expected, the packet arrival ratio of each case with rwalk/ob is almost the same. On the other hand, we can see the difference in UPF mobility. The difference becomes small as the number of nodes increases, but still remains in the case of 80-90 nodes. The average of the hop length in Fig. 2.14 is small when the number of nodes is also small because it is hard to establish a route with many hops. When the number of nodes is larger than 50, base station C placed at the center of the field provides the shortest length of all the placements in rwalk/ob. Although, in UPF, there is a little difference between the hop length in the case of the base station C and that of the base station A placed at the upper right. It is known that as the density becomes higher, a route is shorter because one hop length can be longer in high node density situation. In addition, there are many nodes near the base

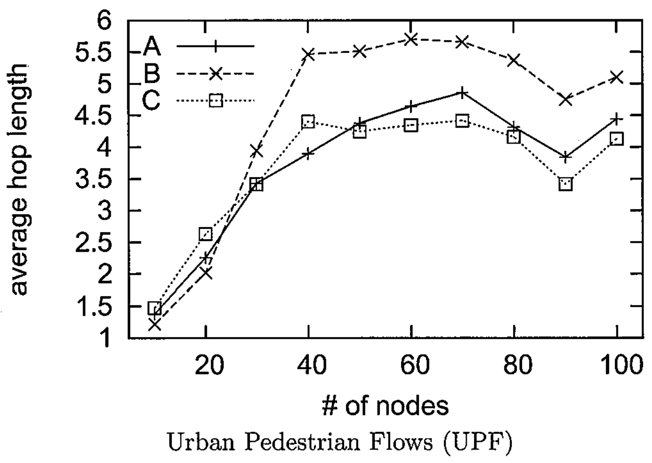


Urban Pedestrian Flows (UPF)

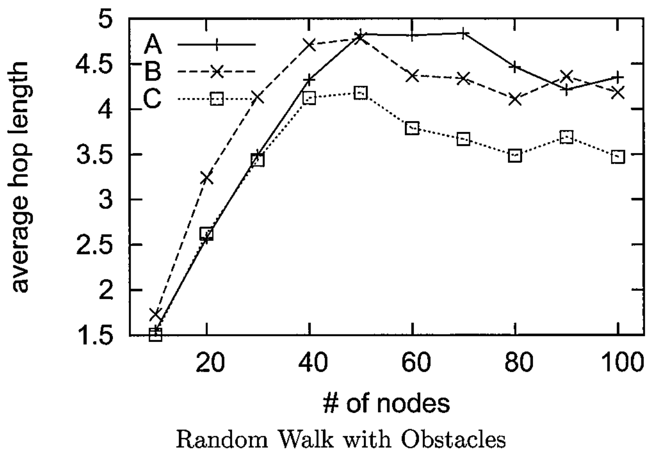


Random Walk with Obstacles

Figure 2.13: Packet arrival ratio.



Urban Pedestrian Flows (UPF)



Random Walk with Obstacles

Figure 2.14: Average hop length of DSR route.

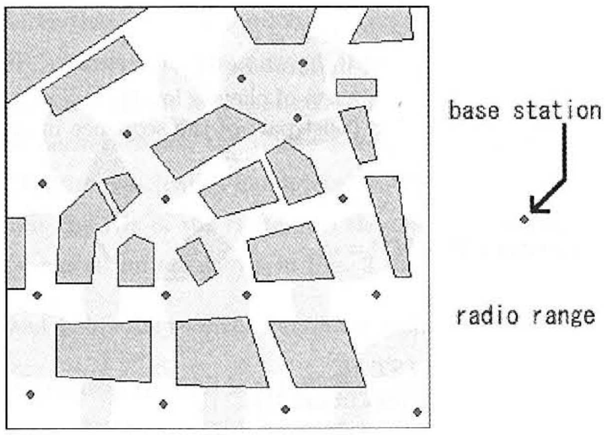


Figure 2.15: Placement of mesh routers.

station A so that the randomly chosen client walks near the base station A with a higher probability in UPF than in rwalk/ob. From these results, we found that base station C in the middle-density section could achieve almost the same packet arrival ratio as that of base station A in the high-density section, and that base station A could provide similar hop length with base station C in the center of the field.

Consequently, distribution of the density may influence the performance of the location-dependent applications such as location planning of base stations. It is necessary to use a mobility model that can reproduce realistic position distribution like the UPF model to evaluate the performance considering the characteristics of the target field.

2.4.2 Prediction-based Mobility Management in Wireless Mesh Network

As another example, we deal with mobility management for wireless mesh networks composed of a set of base stations. Each client connects to one of the base stations (mesh routers) to utilize the network. The radio ranges of a client and a router were 100 and 200 meters, respectively. The placement of mesh routers is shown in Fig. 2.15 where polygons represent obstacles and others are walkways. The mesh routers were placed to cover almost all the pedestrians' walkways.

```
a set of router sequence $HLIST = \phi$;
for( each movement log $A$ in database ){
  if( $A$ includes same sequence of client's log ){
    $HLIST = HLIST \cup$ {post part of the sequence in $A$};
  }
}

a set of router $CR, TMP = \phi$;
do{
  $CR = CR \cup TMP$;
  $TMP = \phi$;
  for( each $A$ in $HLIST$ ){
    $TMP = TMP \cup$ {pop_front($A$)};
  }
}while( $\#(CR \cup TMP) \leq K$ ){

the candidate routers = $CR$;
```

Figure 2.16: Algorithm to select candidate routers.

The basic concept of mobility management schemes developed for cellular or mobile IP networks can be applied to wireless mesh networks [37]. In this experiment, we evaluate the efficiency of the prediction-based mobility management protocol that reduces the location updates by predicting the current location of mobile nodes from past locations. There are many protocols that utilize mobility prediction for mobility management [35, 38, 39]. The protocol of this experiment predicts mobility by the approach based on data mining like [36].

Each client holds its movement log as a sequence of neighbor routers like "$r_1, r_2, ..., r_n$". we assume that the client can recognize the neighbor router by a hello message or traffic for other clients. We also assume that each mesh router has enough movement logs of clients collected beforehand in its database. In this protocol, the client sends its movement log to the router in the location update process. The router predicts the movement of the client using its database. The client and the router utilize the prediction result for the next location update and paging, respectively.

The location update initiated by the client consists of the following steps.

1. First, the client that executes the location update sends a location update packet including its movement log to the neighbor router.

2. The router that receives the location update packet holds the client ID (e.g. IP address) and the current time. Then, the router searches for its holding movement logs that contain the same sequence as that in the received packet. Next, the router selects up to K routers to which the client likely moves (called candidate routers and K is given in advance) and sends the IDs of the routers to the client. The details of the router selection algorithm are shown in Fig. 2.16.

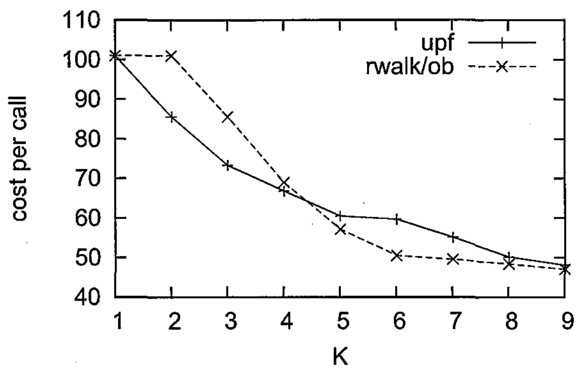
3. The client holds the received candidate router IDs.

If the client recognizes that he is entering an area covered by a non-candidate router, then the client initiates a location update. The router also holds the candidate routers of the client. If a request for a call was received, then all the areas of the candidate routers of the callee are paged at almost the same time.

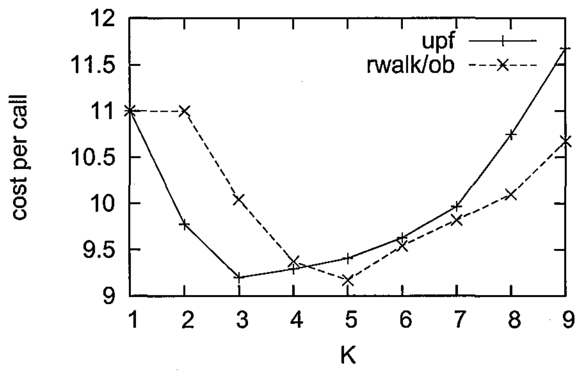
In the experiment, we simulated the performance of this protocol in the UPF mobility and in the Random Walk with Obstacles (rwalk/ob). We adjusted the node lifetime of rwalk/ob so that the distribution of the lifetime can almost be the same as that of UPF mobility. Other simulation settings are the same as in Section 2.4.1.

We calculated the total cost per call arrival. Here, U is the cost for paging the area of a router and V is the cost for a location update. Call-to-mobility ratio (CMR) represents the relative frequency of call rate and mobility and is defined as "an average stay time in the area that a router covers" divided by "an average interval of the call arrival". Figure 2.17 (a) shows the cost per call for U=10, V=1 and CMR =0.1 as applied in [35, 40]. The cost is smaller in UPF than in rwalk/ob for $K \leq 4$. The possible flow of the node is limited in the UPF so we can limit candidate routers. On the other hand, rwalk/ob has smaller cost for $K \geq 5$. Nodes in UPF go far away in a short time because they walk along the shortest path, but nodes in rwalk/ob decide movement randomly so some of them may return to the same point. This difference seems to affect the total number of the location update. Figure 2.17 (b) shows the cost per call for U=5, V=1, CMR=0.5. K (the maximum number of the candidate routers) that has the smallest cost differs in the two mobility models.

Consequently, from these results, the movement routes of the nodes have an influence that may not be negligible in the performance evaluation, especially



(a) U=10, V=1, CMR=0.1



(b) U=5, V=1, CMR=0.5

Figure 2.17: Cost per call arrival.

of the mobility-aware protocols. We should carefully choose the mobility model considering this influence.

2.5 Conclusion

In this chapter, we introduced a realistic mobility model called Urban Pedestrian Flows. By the UPF model, we can reproduce realistic pedestrian flows from data obtained by simple observation such as fixed-point observation using digital cameras or web cameras. To my best knowledge, no existing method considers this research direction. We showed the modeling example of the UPF model and the differences of characteristics between the random-based mobility and the

UPF mobility are shown by the experiment described in Section 2.3.3. Also, through two case-studies, we showed that we could design several MANET protocols and applications and verify their performance on the network topology, which was created by the UPF model and had never appeared in the existing random-based mobility models.

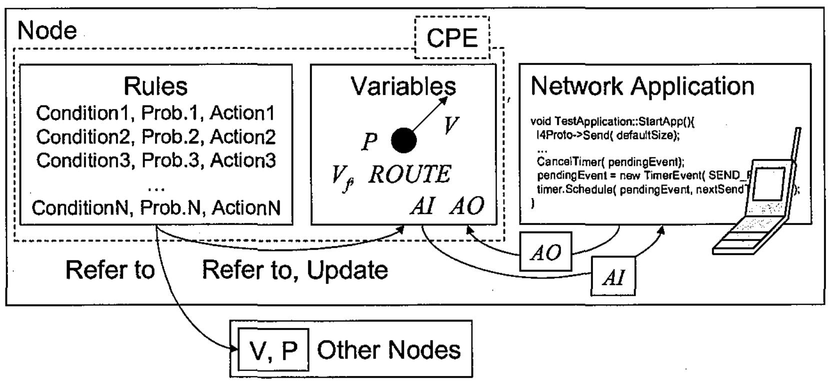
Chapter 3

Condition Probability Event Model

In this chapter, we introduce Condition Probability Event (CPE) model to describe the relational and dynamic behavior of pedestrians toward surroundings, such as adjusting walking speeds and directions to avoid collision with neighbors and obstacles, and stopping at a traffic signal. The key function of the CPE model is the interaction between the network application and the node mobility. In Section 3.1, the details of the CPE model are explained and a description example is also shown. we show an evaluation example of the information diffusion protocol with MANET in which the movement paths of some nodes are changed depending on the network application status by the CPE description in Section 3.2. The topics related to the implementation of the CPE model are described in Chapter 4.

3.1 Condition Probability Event Model

The UPF model focuses on macroscopic movement of nodes, while the Condition Probability Event (CPE) model focuses on the microscopic behavior of the individual nodes. The CPE model allows us to dynamically and more precisely control the behavior of nodes on pedestrian flows determined by the UPF model. With this model, we may describe such behavior that a mobile node makes a detour (changes to another pre-scheduled route) when it receives traffic jam information through MANETs or cellular networks.


Node
CPE
Rules
Condition1, Prob.1, Action1
Condition2, Prob.2, Action2
Condition3, Prob.3, Action3
...
ConditionN, Prob.N, ActionN
Variables
P
V
V_f ROUTE
AI AO
Network Application
AO
AI
Refer to
Refer to, Update
V, P
Other Nodes

```
void TestApplication::StartApp(){
  l4Proto->Send( defaultSize);
  ...
  CancelTimer( pendingEvent);
  pendingEvent = new TimerEvent( SEND_
  timer.Schedule( pendingEvent, nextSend
}
```

Figure 3.1: Interaction between CPE instance and network application.

We would like to note that general-purpose programming languages such as C or Perl allow a wide variety of descriptions, and therefore can be used to describe the detailed behavior of nodes. However, this generality may confuse designers. Thus we propose the CPE model for simulator users to simply describe the behavior of nodes (simulation scenarios). In the CPE model, the next action of a node is determined by its current status (position, direction, velocity, scheduled route and so on) and environmental factors (global time, information from network systems and so on), and this decision can be probabilistic. This model captures well humans' decision-process of actions (action is probabilistically determined by current status and environment), allowing reasonable abstraction of unnecessary behavior for the fidelity of modeling. Thus it has much better applicability for behavior modeling than general languages.

We depict my concept of modeling pedestrian behavior in Fig. 3.1. Formally, the CPE model consists of a list of rules where each rule is a tuple of a *condition*, a *probability* and an *action*, and *node state variables*, which represent the status of each node and are introduced later. The rules can refer to and update node state variables like velocity vector and position of the node. We may specify the same set of rules to all nodes or a different set of rules to each group of nodes. We specify a logical formula as a condition, a value in the range of [0,1] as a probability and a set of substitution statements that may update values of the node state variables as an action. For each increment of the simulation clock (referred to as T), all rules satisfying the given conditions are executed one-by-one from the top of the list with the specified probabilities.

Each node has six node state variables: current position P, scheduled route $ROUTE$ (sequence of vertices), base velocity V_f, actual velocity vector $\overrightarrow{V}$, input data AI of the node to the network system (application input) and output data AO from the network system to the node (application output). These variables are referred to and updated by the rules. When a node is generated according to the UPF scenario derived in Section 2.2, an instance of the CPE model of the node is initialized as follows. The path $r_k \in R$ is set to the initial value of route $ROUTE$, and the velocity v_0 in Exp. (2.2) is set to V_f, which represents the velocity in a sparse area. Then normal descriptions in the CPE model contain such rules that update $ROUTE$ and V_f followed by the rules that update current position P and actual velocity vector $\overrightarrow{V}$ accordingly. These rules determine the basic behavior of nodes that walk along the routes. The actual velocity $|\overrightarrow{V}|$ cannot be greater than the base velocity V_f, but may be smaller when the node cannot walk smoothly in a crowd.

The CPE model assumes that an application program of the network simulator is notified of the value of application input AI, and updates the value of application output AO. In the rules of the CPE model, we can set any value to application input AI but can only read application output AO. The variables AI and AO enable simulator users to describe interactive behavior of nodes with the application program. We note that in order to enable the application program and the CPE instance to interpret AI and AO, several libraries are offered to mitigate the developers' effort.

We offer many pre-defined functions to help the developers to specify realistic behavior of nodes in the CPE model. For example, we provide a function that returns the positions and actual velocity vectors of nodes within eyesight. A part of them are presented in the following example.

3.1.1 Description Example

Here, we consider the following MANET system. Shoppers holding information terminals with short range radio communication devices exchange useful information (e.g. sales information) when they encounter each other.

We present a description written in the CPE model for the shoppers in Table

Table 3.1: Example behavior description of shoppers in CPE model.

	Condition	Prob.	Action
E1 (exp.)	AO == "SHOP_A" (receives string "SHOP_A" from the application)	0.30	ROUTE = shortest_path(P, Point(SHOP_A)) (change destination to vertex SHOP_A)
E2	P == Point(SHOP_A) (arrives at vertex SHOP_A)	0.10	AI = "SHOP_A" (give an application input "SHOP_A")
E3	(P == Point(INT_1)) $\wedge$ ((T%2min) < 1min) (stops at vertex INT_1 with traffic light which changes every minute.)	1.00	stop(); (wait until the light changes to green)
E4	– (execute always)	1.00	V = UPFvector(V_f, P, ROUTE, neighbors(P, V_f)); (calculate velocity vector from state various variables)
E5	– (execute always)	1.00	P = UpdateLocation(P, V); (update position)

3.1. Rule E1 describes the behavior that changes the scheduled route *ROUTE* dynamically, according to an application output *AO*. If the shopper receives information "SHOP_A" given by the application, he/she changes his/her destination to the vertex "SHOP_A" with probability 0.3. This is done by updating the *ROUTE* variable to the shortest path from current position *P* to the vertex specified by Point(SHOP_A). Rule E2 describes the behavior that gives an input to the application depending on the position of the given node. More concretely, the node gives an application input "SHOP_A" to variable *AI* with probability 0.1 when he/she reaches the vertex specified by Point(SHOP_A). We note that the application is programmed to distribute the given input to the neighbor nodes. Rule E3 implements the behavior that follows a traffic signal at intersection INT_1. A function "stop()" used in the action part sets 0 to the node's base velocity V_f temporarily. The condition of E3 becomes true if the simulation time T (min.) is even, which means the period of the red light.

Rules E4 and E5 define fundamental behavior based on the UPF mobility. Rule E4 calculates velocity vector $\vec{V}$ from current position P, route $ROUTE$,

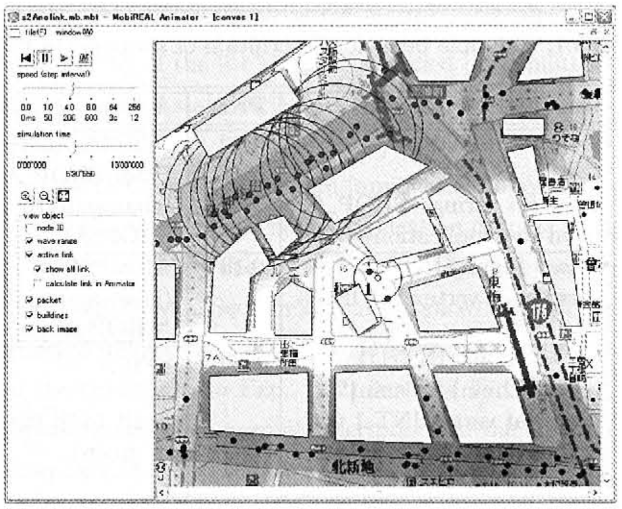


Figure 3.2: Snapshot of information diffusion scenario.

base velocity V_f and the position of the neighbor nodes $neighbors(P, V_f)$. We note that we have modeled evasive actions (turn or slow down) against the approaching people. The UPFvector() in Rule E4 calculates velocity vector $\overrightarrow{V}$ based on this model to make the behavior of the node more realistic. Rule E5 updates current position P based on velocity vector $\overrightarrow{V}$ calculated by Rule E4.

3.2 Case Study

We have carried out some experiments to investigate the influence of the mobility models on the performance evaluation of MANET and to show the usefulness of the CPE Model.

We have simulated an information diffusion system which disseminates advertising information such as sale and happy hour information to mobile nodes on MANET. In this experiment, we have investigated the impact of dynamic behavior change of nodes depending on the information received from the network application, by comparing two cases: with and without dynamic behavior change.

If a mobile node executing this application receives information broadcasted by a short-range base station or a neighboring node, it caches the information.

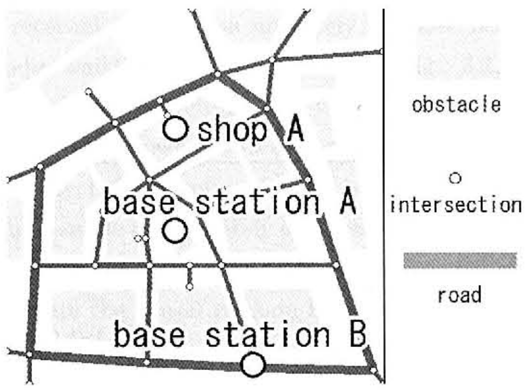
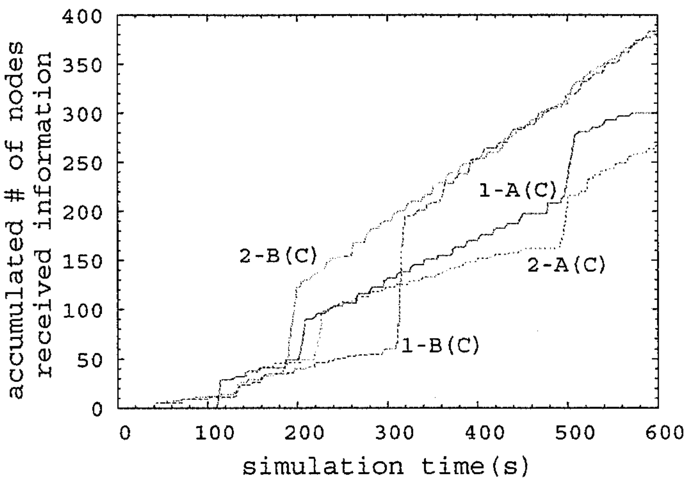


Figure 3.3: Downtown Osaka city and base-stations.

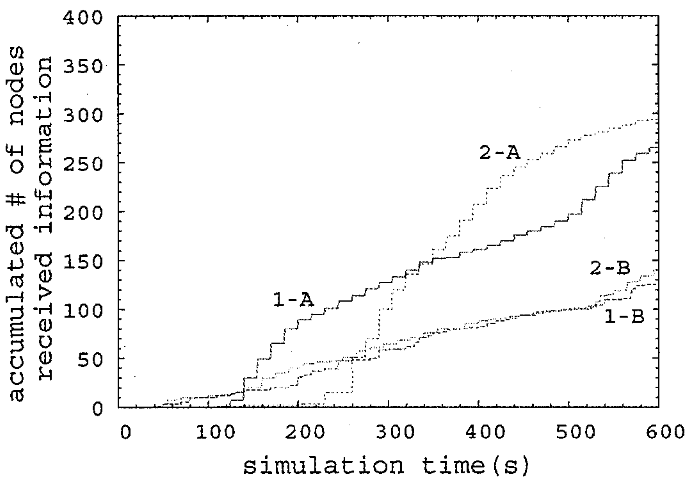
After that, for every 15 seconds, it checks whether it should broadcast the cached information to its neighbors or not, according to the predefined probability function (*diffusion policy*) including Euclid distance $d(P, S)$ between P and S where P and S denote the positions of the node and shopA (Fig. 3.3). The mobile node discards the cached information after 60 seconds or once it broadcasts the information. We have used the following two diffusion policies, (i) $prob. = 1 - \frac{d(P,S)}{500\sqrt{2}}$ (if the node is located closer to the shop, it broadcasts with higher probability), (ii) $prob. = \frac{d(P,S)}{500\sqrt{2}}$ (this is the opposite of the diffusion policy (i)). And we have used either one of the two base stations A or B, respectively (Fig. 3.3). We have had 4 scenarios by the combinations of the diffusion policies and base stations. We name the scenario for the pair of a diffusion policy i and a base station x as scenario i-x. For each scenario i-x, we have also had the corresponding scenario named scenario i-$x(C)$ in which the node changes its destination to shop A when the node receives information.

The simulation time is 600 sec., and after 180 sec., information is diffused for every 2 sec. from a base station. We used IEEE802.11 DCF with RTS/CTS as the MAC protocol. The radio range was set to 100 meters. The total number of nodes was 2000 and 20% of them used the application. The speeds of pedestrians were randomly selected between 1.1m/s and 1.7m/s.

We show the number of nodes which received information in Fig. 3.4. In the scenario of base-station B, we see a big difference between two cases with and without dynamic behavior change. With dynamic behavior change, some



(a) with dynamic behavior change



(b) without dynamic behavior change

Figure 3.4: Accumulated number of nodes that received information.

of nodes which received information are likely to move near the shop A. So, a part of the nodes walking in the upper side of Fig. 3.3, which is far from the base-station, could receive information via the nodes near the shop A. Dynamic behavior change as shown in this case study is difficult to predict without simulation. Especially, it is difficult to predict how many nodes change their behavior. So, it is impossible to accurately simulate the realistic behavior of nodes with dynamic behavior change only by the recorded behavior of nodes. In order to accurately simulate the causal relation between the ratio and the influence to the network system, the proposed simulation method would be essential.

3.3 Conclusion

In this chapter, we introduced a microscopic mobility description model called Condition Probability Event. By the CPE model, we can reproduce the interaction between MANET applications and behavior of nodes. We have shown that dynamic behavior change reproduced by CPE affects the performance of the MANET system through a simulation example. The CPE model requires the interaction between the mobility simulation module and the network application simulation module. The CPE model is integrated into the MobiREAL simulator and topics about the implementation are described in Chapter 4.

Chapter 4

Architecture of the Wireless Network Simulator MobiREAL

In this chapter, the architecture of the MobiREAL simulator is described. The simulation module of the MobiREAL is divided into two programs, the behavior simulator and the network simulator, considering the applicability to existing network simulators. Two programs run in parallel and exchange simulation results to simulate the relational behavior between the network status and the node mobility specified by the CPE model. The MobiREAL simulator enables the use of real application codes on real mobile terminals connected to a simulated network. The MobiREAL simulator contains a GUI tool which manipulates the movement of mobile nodes, gives the user inputs to the target application in run-time, and visualizes the node mobility and network status during the simulation. Using these functions, for example, we can test to what extent video quality decreases in multicasting movies on MANET. We show an experiment of the performance of the MobiREAL simulator in Section 4.3 and two case-studies, the evaluation of the DSR routing protocol using the run-time operation function and VoIP communication using real terminals, in Section 4.4.

4.1 Related Work

Network simulators such as ns-2 [26], OPNET [41] and Qualnet [42] have been widely used for the evaluation of network traffic. On the other hand, to test and evaluate the end systems themselves, many network testbeds have been developed [43–50]. StarBED [43] is a large-scale network testbed integrating simulation with emulation. They offer hundreds of physical nodes and can emulate thousands of virtual nodes. ORBIT [44] also offers about 400 inter-connected stationary wireless nodes that can emulate specified network topologies and traffics. Due to their features, they are mainly used to test the wide-scale (e.g. campus-scale, city-scale or Internet-scale) networks.

On the other hand, other types of testbeds like [47, 49–51] can be built in small spaces such as laboratory rooms and thus can easily be used to test end system applications. They mainly use physical nodes with some emulation techniques. In MobiEmu [51], using a packet filtering technique at the MAC layer, a testbed for implementing a given network topology is constructed. Mobile Emulab [49] takes a direct approach that uses mobile robots to move terminals physically. In MiNT project [50], a miniaturized wireless network testbed is constructed. In these testbeds, the accuracy of packet transmission is a benefit, however the scale and topology of target networks are limited by the number of physical nodes, their capability of movements, and space to deploy these nodes.

The hybrid approach in which simulation is integrated into emulation is reasonable to balance the timeliness of packet processing, adaptivity to the scale and topology, and hardware cost. ns-2 [26] implements an emulation function that allows real nodes to connect to simulated networks. TWINE [46] is a hybrid emulation testbed for wireless networks and applications, which can adaptively combine simulated, emulated and physical subnets. Based on this, a WHYNET framework is presented in [52] and several case studies are given. MobiNet [48] enables to test IP-based unmodified applications by emulating multi-hop wireless networks over a LAN cluster.

My approach is close to the hybrid approaches, however it differs in the following points. We mainly focus on highly dynamic mobile wireless networks where node mobility strongly affects the network performance [1, 2, 53]. In such a network, application developers may want to test application performance with many mobility scenarios and geographical environments. To help those

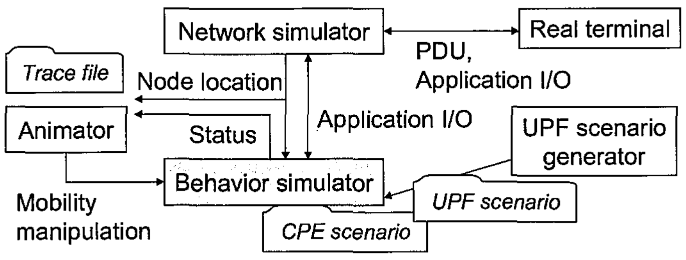


Figure 4.1: MobiREAL simulator overview.

developers, we provide such a system that can simulate the realistic behavior of nodes and can control the movement of nodes. Also these operations can be done through GUIs. By this feature, developers can easily generate intentional route disconnection in testing routing protocols and many other situations that are hard to predict in advance. We have presented the results from case studies considering real-world applications.

4.2 MobiREAL Simulator

We have developed a network simulator called *MobiREAL* to simulate MANET systems with the realistic behavior of nodes based on the UPF model and the CPE model.

The architecture of the MobiREAL simulator is shown in Fig. 4.1. MobiREAL is composed of a network simulator that simulates network systems, a behavior simulator that simulates the movement of nodes, a UPF scenario generator and an animator. Also, real teminals can be connected to the MobiREAL Simulator and details are described in Section 4.2.1. The behavior simulator takes three inputs, a simulation field, a UPF scenario and a set of CPE scenarios. As a simulation field, we specify the structure of streets and buildings that interfere with radio propagation. As a UPF scenario, we specify the routes of nodes and their node generation rates (flow rates). According to the UPF scenario, MobiREAL generates node instances that move based on the CPE scenarios. With the UPF scenario generator, we can generate a simulation field and the corresponding UPF scenario graphically and intuitively. Application layer protocols in the network simulator may contain the description for the CPE scenario to interact with mobility simulation.

Since the simulator part of MobiREAL is composed of two independent programs (i.e., behavior simulator and network simulator), we have implemented MobiREAL according to the following concept. First, both simulators hold the positions and speeds of nodes independently and synchronize the progress of simulations. The behavior simulator calculates the latest positions, directions and speeds of nodes, and sends them to the network simulator. In response, the network simulator updates its information about the nodes according to the received data, sends outputs of the application to the behavior simulator and continues the simulation. The behavior simulator can perform dynamic behavior change according to this information.

4.2.1 Cooperation with Real Terminals

In the MobiREAL Simulator, real mobile terminals are connected via low delay networks (*e.g.* LAN) to a single host that simulates its network behavior. The real terminal is mapped to one node in the simulation. Hereafter, we call the node corresponding to the real terminal *real nodes* and others *simulated nodes.* All nodes appear in the simulation field. Application codes and higher layer protocols of real nodes are executed on real stations and lower layer protocols and radio models are simulated. Applications and protocols on real nodes use MobiREAL APIs to interact with simulated lower layers of the node instances assigned to those real terminals so that they can communicate with simulated nodes and vice versa.

MobiREAL uses TCP communication with the socket library for cooperation between real terminals and the network simulator. It does not modify its OS dependent libraries nor limit the environment of real terminals. We offer three layers' APIs, the application layer API, the transport layer API and the network layer API. We specify one of the above APIs used by each real terminal. If a protocol data unit (PDU) of the specified layer is transmitted as data of the socket communication from the real terminal, the PDU is translated into the usable format in the simulator and is passed to the corresponding layer of the assigned node instance. In the case of using the application layer API, the real application sends the destination IP address, type of transport layer protocol, data to be transmitted and so on to the simulator, then the simulation of the

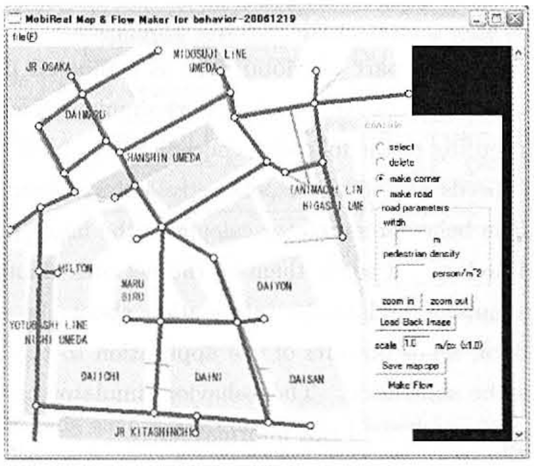


Figure 4.2: GUI support for map and UPF scenario generation.

transmission is initiated from the transport layer. If the node instance assigned to the real node receives a PDU of the specified layer, the PDU is transmitted to the corresponding real terminal.

4.2.2 UPF Scenario Generator

We have developed the *MobiREAL UPF scenario generator*. Showing a BMP graphic map file like Fig. 4.2, simulator users can specify a street graph through the GUI with their mouse devices. We specify two attributes (width and observed node density) for the street. This tool also assists in generating potential candidate routes as described in Section 2.2.1. Then this tool generates a program code of the UPF scenario that can be run in MobiREAL where the linear programming problem solver *lp_solve* [34] is used to calculate the flow rate on each route.

4.2.3 Network Simulator

The network simulator needs to synchronize with the behavior simulator to exchange information, update the positions and the speeds to follow the behavior simulator, and process application inputs and outputs. We have developed the network simulator by extending and modifying GTNetS [6] developed at the Georgia Institute of Technology. GTNetS aims at improving the scalability of large-scale simulation, so it has a mechanism for parallel simulation of wired networks. For simulation of wireless networks, GTNetS also supports routing protocols like DSR, AODV and NVR (wireless form of Nix-Vector routing) and standard MAC and physical layer protocols like IEEE 802.11.

In GTNetS and many other simulators, the movement of nodes is bounded by boundaries, or a node moving through a boundary appears from the other side. This means that the number of nodes is fixed throughout simulations. On the other hand, my mobility requires dynamic generation and deletion of nodes. In our MobiREAL network simulator, we have modified the node management function of GTNetS to implement the dynamic generation and deletion of nodes. Also, we have modified the simulation module of the physical layer to implement radio propagation attenuation under the existence of obstacles in order to achieve more realistic wireless network simulation. In this radio propagation model, attenuation when the radio penetrates obstacles is considered. For this purpose, simulator users can specify an attenuation coefficient for each obstacle.

4.2.4 Visualization and System Control

MobiREAL provides a set of control and visualization interfaces (Animator in Fig. 4.1) which have the following three functions: (1) visualization of simulation status, (2) run-time manipulation of node movement, and (3) remote operation of applications on real terminals. We illustrate how multiple components interact to provide these functions in Fig. 4.3.

The Animator runs on the Microsoft Windows platform. The Animator and the simulator cooperate by exchanging control data over TCP connections. Therefore, the simulator and the Animator can run on separate computers. For example, we can execute the simulator on a high-performance workstation and the Animator on a consumer PC.

Visualization

The MobiREAL Animator visualizes the traces of the simulation. Visualization of the simulation results is very useful for designing, debugging and presenting network systems. In particular, visualizing mobility of nodes is mandatory to confirm their realistic behavior.

The Animator can animate the movement of nodes, the topology of wireless

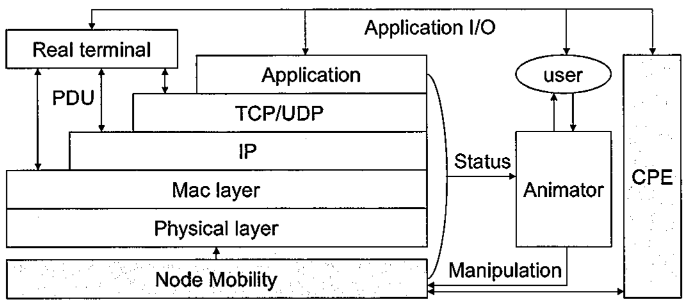


Figure 4.3: Logical structure of the MobiREAL simulator.

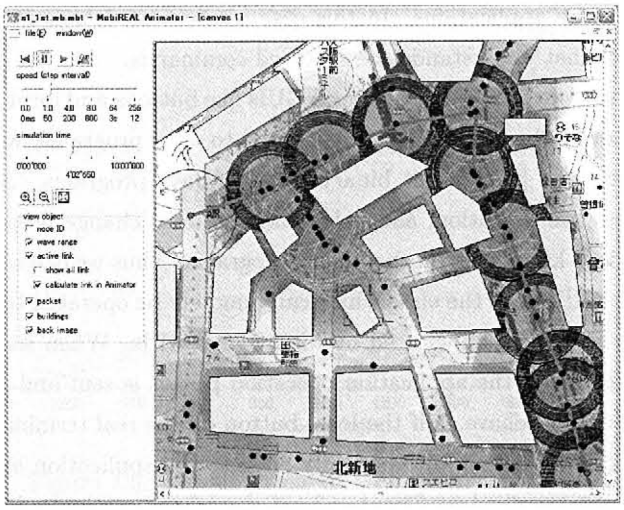
links, packet propagation and so on. We can choose whether these objects are to be displayed or not. Each mobile node is represented as a small circle, and a semi-transparent concentric circle represents its radio range. Each packet transmission is drawn by concentrically growing circles with colors. First, a circle with 1 pixel radius is drawn when a packet is sent. Then the radius of the circle gradually grows until it reaches the radio range of the node sending the packet. The color of nodes can be freely set in a network application program. For example, the Animator can show the nodes that received certain packets in red and others in blue. A snapshot from the Animator is shown in Fig. 4.4 (a).

The Animator can also show some measured metrics like node density and packet loss rate as shown in Fig. 4.4 (b). In this function, a simulation field is separated into grid cells, and they can be color-coded according to the measured values. Thus users can analyze, for example, the influence of geography on the density of nodes, and the relationship between the density of nodes and network performance. For interested readers, several movies and snapshots of the Animator are presented on the MobiREAL web page [54].

Operation of Mobility/Application

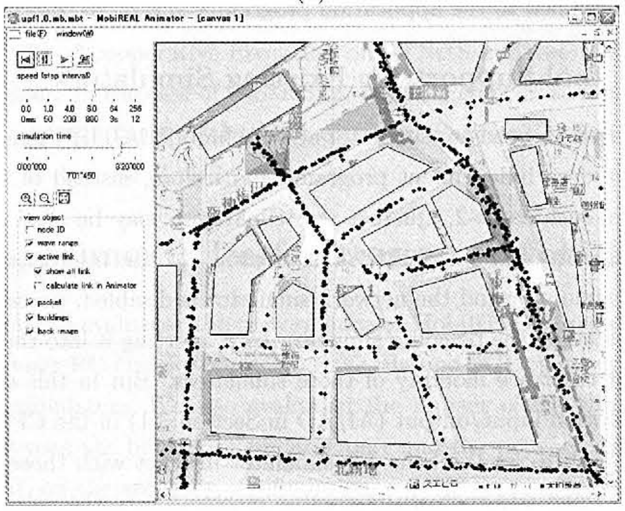
We can control movement of nodes manually through the Animator. We can change the positions, speeds and directions of nodes and add/delete nodes. In MobiREAL, the movement of a node is represented as a sequence of *waypoints*, each of which consists of (x,y)-coordinates, pause duration and velocity. During execution, waypoints can be added, deleted, or modified.

For usability, it is desirable to operate applications running on plural real



[ ] node ID
[✓] wave range
[✓] active link
[ ] show all link
[✓] calculate link in Animator
[✓] packet
[✓] buildings
[✓] back image

(a)



[ ] node ID
[✓] wave range
[✓] active link
[✓] show all link
[ ] calculate link in Animator
[✓] packet
[✓] buildings
[✓] back image

(b)

Figure 4.4: Snapshots from MobiREAL Animator.

terminals from a single interface. One way is to let the application programs and the Animator exchange *application operation packets* through the MobiREAL APIs, in parallel with data packet exchange. This packet includes commands to operate the applications. In this case we may need to implement a simple interpreter that understands the specified commands.

Another way is to prepare general GUIs like buttons and forms in the Animator. Although the applications are limited to Java programs, we have decided to use Javassist [55] to edit binary codes of Java programs. Javassist allows to change class definition, add fields and methods, change methods and so on without deep knowledge on the binary programs. Thus we can replace the common class GUIs with the class which can remotely be operated by the Animator through the TCP channel used by MobiREAL APIs. When we click a button on the Animator, the application operation packet is sent and the application program would behave as if the local button on the real terminal were clicked.

In addition, we can automatically operate the application according to the CPE model described in Chapter 3. if one of the methods described in this section is applied to the application program.

4.2.5 Tool Support for Existing Simulators

The MobiREAL behavior simulator and the MobiREAL UPF scenario generator can be used as independent programs. Therefore, instead of GTNetS, other simulators such as ns-2, Qualnet and GloMoSim may be used as the network simulator part of the MobiREAL simulator. If the interaction between the behavior simulator and the network simulator is disabled, we can use the trace file generated by the behavior simulator by converting it into the corresponding inputs for the trace mobility of those simulators. But in this case, we cannot use application input/output (AI/AO in Section 3.1) in the CPE model.

If we wish to let the behavior simulator interact with those network simulators, we must add such an interaction mechanism to the network simulators. Usually, wireless network simulators including GTNetS have a "mobility class," in which behavior of mobile nodes is described. In the case of the MobiREAL simulator, we have enhanced the mobility class of GTNetS so that the network simulator can periodically replace the speeds and directions of nodes with the latest ones received from the behavior simulator. According to this concept, we

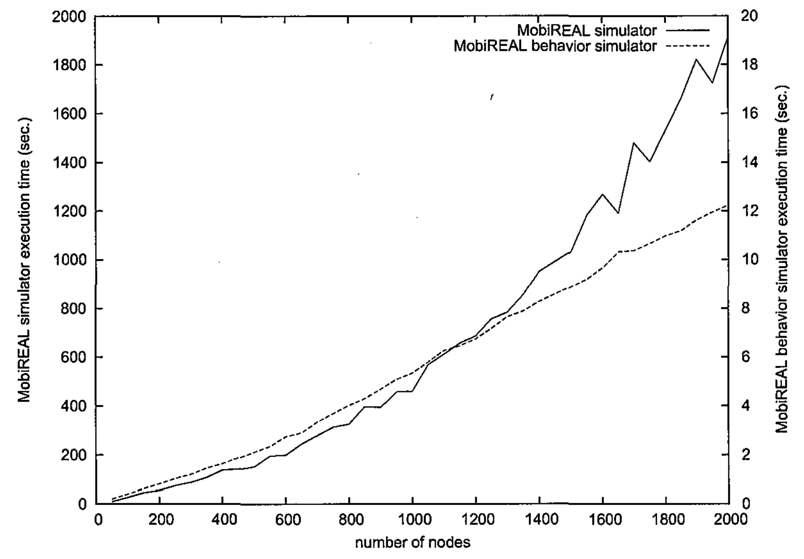


Figure 4.5: Running time of MobiREAL simulator.

have accomplished cooperative execution of GTNetS and the MobiREAL behavior simulator. In the case of GloMoSim, *MobilityTrace* class can be modified in the same way. Similarly, *MobileNode* class and *PositionHandler* class should be enhanced in the case of ns-2.

4.3 Experiment: Performance of MobiREAL

In this section, we evaluated the performance of MobiREAL simulator. we have used a consumer PC (Intel Xeon CPU 3.06GHz and 2GB memory) to execute MobiREAL simulator. We also evaluated the impact of the synchronization overhead between the behavior simulator part and the network simulator part of MobiREAL on the accuracy of simulation.

Running Time Fig. 4.5 shows the actual running time of MobiREAL simulator with interaction between the behavior simulator and the network simulator (Y-axis of the left-hand side) and the actual running time of MobiREAL behavior simulator itself (Y-axis of the right-hand side) to execute the scenario, varying the number N of nodes. The running time of the behavior simulator

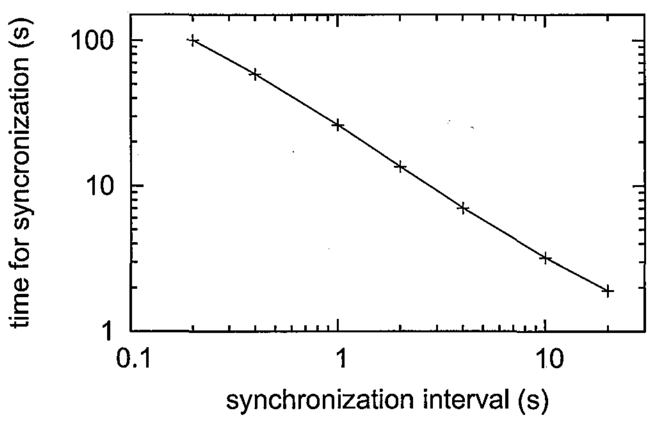


Figure 4.6: Running time for interaction.

increases linearly and is very short, while that of MobiREAL simulator increases more quickly. Usually network simulators take $O(N^2)$ time to compute neighboring nodes and many complicated computation compared with the behavior simulator. Therefore the result in Fig. 4.5 is very natural, and the overhead of introducing my mobility model to the network simulation time is very small. From the viewpoint of scalability, MobiREAL simulator can execute scenarios with over 2,000 nodes. We believe that the capability to handle a few thousands of nodes is enough for simulating large-scale MANET applications.

Time for Interaction and Position Error In this experiment, we evaluated the trade-off between the accuracy of the node position and the synchronization cost. In the simulation of MobiREAL, the behavior simulator holds accurate positions of nodes. The network simulator periodically synchronizes with the behavior simulator, receives the positions and velocity vectors of nodes, and calculates current nodes' positions using received values until the next synchronization. If the synchronization interval is too long, the velocity vector changed by the behavior simulator may cause a large error between the positions held by two simulators. However, if the synchronization interval is too short, the computation cost for the synchronization increases.

We used the mobility scenario derived in Section 2.3.1. The speed of the node followed uniform distribution between 1.0 m/s and 2.0 m/s. The simulation time

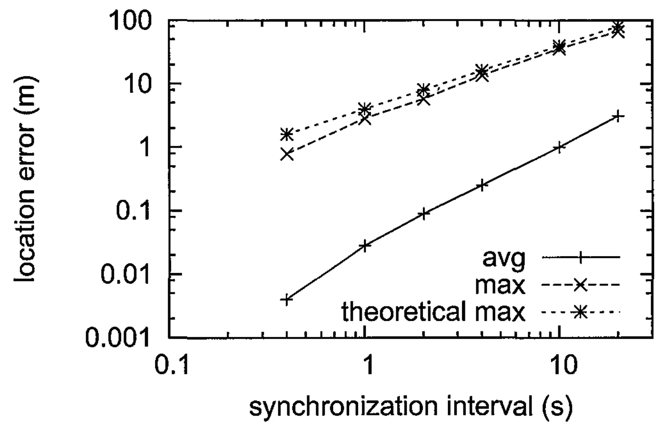


Figure 4.7: Location error between two simulators.

was 100 s and the simulation granularity of the behavior simulator was set to 0.2 s. Results are shown in Fig. 4.6 and Fig. 4.7. Here, the error of the node position is defined as the error between the position which the behavior simulator holds and that which the network simulator holds. The error was sampled just before the network simulator updated its node position by the synchronization.

Obviously, there is a trade-off between the accuracy of the node position in the network simulator (Fig. 4.7) and the total computation time for the synchronization (Fig. 4.6). The error reaches zero when the synchronization interval is equal to the simulation granularity of the behavior simulator (0.2s). For the maximum speed V of nodes and the synchronization interval T, the error must be less than $2VT$ (theoretical limit of the maximal error). So the synchronization interval is considered short enough if $2VT$ is far smaller than the communication range of the node. In my simulations, we set the synchronization interval to one second.

4.4 Case Study

In order to show the usefulness of two features of MobiREAL, which allow us (1) to change the behavior of specified nodes in run-time and (2) to simulate wireless network with real stations, we have conducted some experiments to run

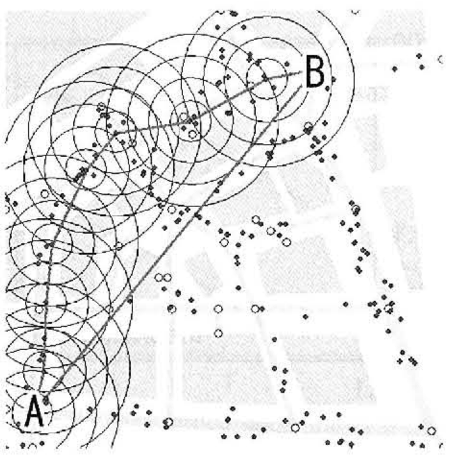


Figure 4.8: Snapshot of visualization tool.

a VoIP application over the DSR protocol on MANET.

4.4.1 Experimental Configurations

We used UPF mobility deribed in the experiment of Section 2.3.1. The size of the field was 500m × 500m. The total number of nodes was about 200 and the speed of each node was 1.1 – 1.7 m/sec. During the simulation, VoIP communication data was transmitted at 9.6kbps. The radio range was set to 100 meters. We used IEEE802.11 DCF with the RTS/CTS mechanism as the MAC protocol.

4.4.2 Intentional Route Break in DSR

In this experiment, we create the situation that was difficult to create without run-time manipulation in order to show its usefulness. We evaluated the recovery time from the DSR route break with the UPF mobility.

In the experiment, CBR traffic is generated from the node at point A in Fig. 4.8 to the node at point B. We stop the simulation temporarily using the GUI of MobiREAL when the route is stable, delete 3 or all intermediate nodes, and restart the simulation. By this we can simulate the situation that multiple nodes disappear at the same time. If we use somewhat complex mobility model to improve the realism, it is difficult to know in advance on which nodes the route is established due to dynamics of network topology and unpredicted movement

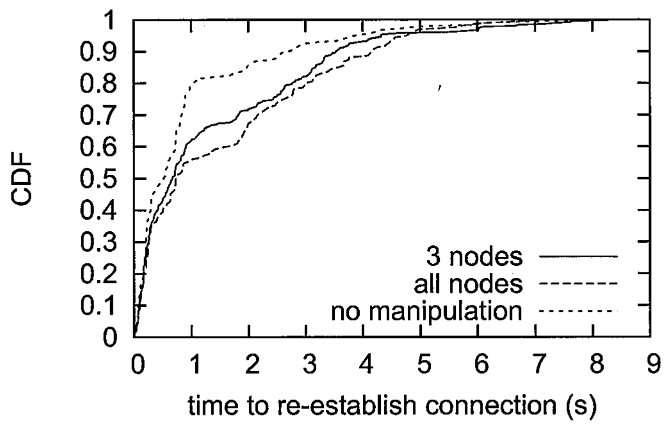


Figure 4.9: Cumulative Distribution of Route Repair Time

of nodes in real environment. To search the intermediate node in run-time, we may have to add some function to the simulator, but we can soon find the intermediate node and can delete the node with GUI of MobiREAL.

Fig. 4.9 shows the cumulative distribution of route repair time. In the graph, "3 nodes" represents the case where three relay nodes on the DSR route were chosen randomly and deleted, and "all nodes" represents the case where all the relay nodes on the DSR route were deleted. Also for comparison purpose, we have evaluated an additional case; "no manipulation" represents the case where no manipulation was applied during simulation. We can see clearly that disappearance of all the intermediate nodes makes the route repair time longer. Also, we can see that the repair time of "no manipulation" is smaller than the other two manipulation cases. From this result, we can expect that in the original node mobility without my operation, such a situation that forwarders suddenly and simultaneously disappear cannot be observed. The run-time operation function of MobiREAL enables us to give some intentional factor into the on-going network simulation according to the visualized current status. Next, we show an example to explain how much MobiREAL helps simulator users in such a case.

To use network simulators, we need certain amount of experience and much effort. To see how MobiREAL helps to alleviate this burden, we have asked three master course students in my laboratory to implement the function that

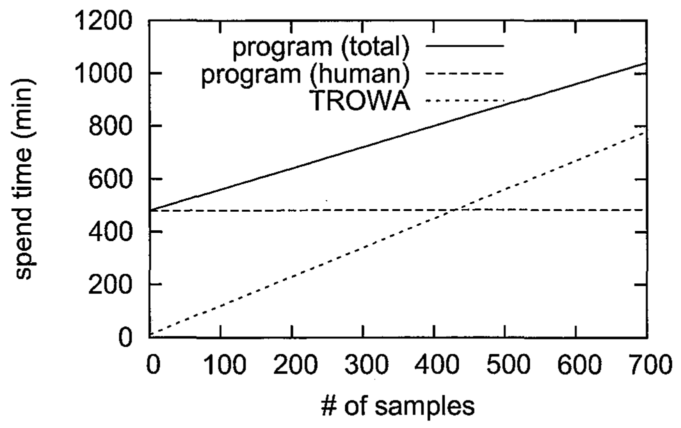


Figure 4.10: Time to collect data of Fig. 4.9.

automates the manipulation in the previous experiment into the network simulator. The function finds the intermediate nodes and deletes those nodes at the same time. The student "A" having an experience of modifying the simulator took about 4 hours, the student "B" took about 12 hours, and the student "C" gave up after 4 hours. It seems to be hard for inexperienced users to understand the source files of the DSR protocol that have about 1600 lines including the header files. Of course, this result is only an example, but one can see that the implementation is not an easy task as everyone can finish in a short time.

Fig. 4.10 shows an example of the time to collect data used to obtain the results in Fig. 4.9 against the number of the collected data samples. We show the time of two data collection methods. One method is to implement the manipulation function as explained above into simulator codes. In this case, we need time to implement the function, which we set to 480 minutes (this is the average time of two students "A" and "B") in addition to the time to run the simulation. "program (human)" in the figure represents the time which we have to spend for the experiment and most of the time is for the implementation. The case "program (total)" represents the total time to collect samples and this includes the running time of the simulation when we can spend our time for other things. Another method is to use the MobiREAL function and is "MobiREAL" in Fig. 4.10. In this case, we do not need to write codes. Instead we directly specify which nodes should disappear through GUI so we have to spend our time while running the simulation.

Even though MobiREAL offers GUIs, we need to operate the GUIs at any moment when we need to change situation. Therefore, it might take much effort if we need to collect large amount of data. For example, as shown in Fig. 4.10, the time to collect samples by MobiREAL exceeds the time required to prepare the program if we collect 420 or more samples. However the advantage of MobiREAL is that we can easily simulate application/protocols under various situations. For example, before we conduct the detailed performance evaluation, we may want to test the application with a wide variety of configurations and find the best configuration in which we can examine the application's various aspects. In general preparing all of the configurations may take long time and require a lot of efforts, thus we provide an environment where we can set and modify configurations through UI observing visualized simulation status. Consequently we may reduce the cost of the trial and error process required in conventional simulations. This advantage can improve the efficiency of the design and the development process.

4.4.3 Voice Quality over VoIP

In order to show the effectiveness of using real application programs in terms of utilizing users' perception for the design of the application, we have asked six students to make conversations through the real terminals in the context of the VoIP application.

It is a common technique for VoIP client software to buffer the voice data to salvage the delayed packets. When we set the longer time for buffering, the data loss ratio becomes smaller and thus the playback quality of the voice will be better, whereas the end-to-end delay becomes larger and the perceptive delay quality will be worse. There is a trade-off between playback quality and delay depending on buffer size. In the experiment, changing buffer size in the application program, we measured the loss ratio of voice data and obtained subjective feedbacks from the students.

Fig. 4.11 depicts frequency that voice data was not received for the interval longer than the specified time duration (0.25 sec to 4 sec in x-axis) on the 8 hop route varying the buffer size of the VoIP application. The figure shows that the buffering can clearly reduce the frequency of longer intervals with no voice.

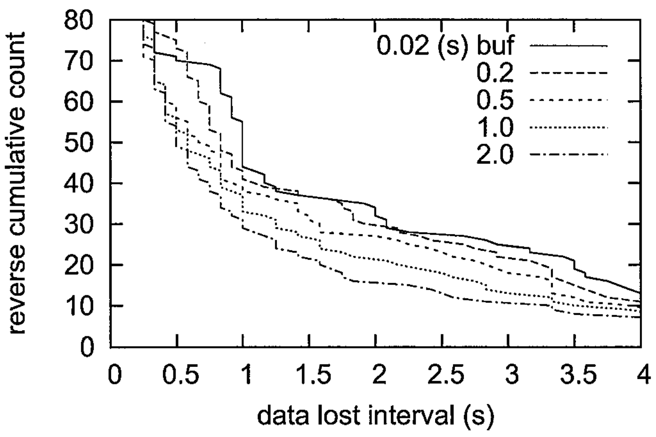


Figure 4.11: Distribution of data loss interval.

Table 4.1: Perceptive voice quality depending on buffer size

buffered time (s)	0.02	0.2	0.5	1.0	2.0
Student A	1	1	2	4	4
Student B	1	1	1	4	4
Student C	1	1	1	3	3
Student D	1	1	2	3	4
Student E	1	1	4	4	3
Student F	2	5	5	4	3

1: terrible, 2:bad, 3:tolerable, 4:acceptable, 5:good

Thus, the playback quality should be improved.

To confirm the degree of this improvement, we conducted a questionnaire to six students where each student played back voice transmitted through 8-hop connection, under five settings of the buffering time. The results are shown in Table 4.1. Most students recognized the improvement of the voice quality when the buffering time was 1.0 or larger. Two students preferred 1.0 sec buffering time than 2.0 sec due to the trade-off between the delay and the playback quality.

Although we can measure traffic characteristics like data loss ratio and end-to-end delay by network simulators, it is difficult to grasp human perception from the measured values. In the above experiment, the appropriate value for the buffering time strongly depends on the user's perception. Therefore, it is quite important to evaluate such real-world performance and develop applications considering it.

4.5 Conclusion

In this chapter, we introduced the architecture of the MobiREAL simulator. In MobiREAL, application programs on real mobile terminals can be tested and verified through the simulated network. The MobiREAL simulator has two simulators, the behavior simulator and the network simulator. The implementations which enable the run-time operation of node mobility and the interaction between the network simulation and the mobility simulation required by the CPE model are explained. The application problems to the existing network simulators are also described in this chapter.

Through the experiment, we showed the trade-off between the computational cost for the simulation and the preciseness of the simulation by changing the interaction period of the behavior simulator and the network simulator. The usefulness of the MobiREAL simulator was shown by two case-studies, the evaluation of the DSR routing protocol using the run-time operation function and the evaluation of the VoIP communication application with real terminals.

Chapter 5

Real-time Simulation

In this chapter, we introduce the proposed real-time simulation technique by prioritizing some simulation events. In Section 5.1, first we explain how to realize real-time simulation and then explain the details of proposed method. In the MobiREAL simulator introduced in Chapter 4, network simulation events are processed prior to the mobility simulation events using this prioritizing method. Section 5.2 shows the experiments which evaluate the performance of real-time simulation and influences of the proposed method.

5.1 Event Prioritizing for Real-time Simulation

For real-time simulation, it is required to synchronize the simulation clock with the real clock, and each simulation event like packet transmission must be executed at accurate time. In parallel, TROWA simulates node mobility, however this mobility simulation may increase its processing load and may have bad influence on real-time simulation. Many realistic mobility models and some random-based mobility models need periodic updates of all nodes' positions. This position update does not require so much computation power. However, if this update is scheduled to the same time with packet processing events, they may cause temporary delay of event processing and we may lose timeliness.

In order to solve this problem, we propose a method to reduce the temporary delay of the simulation clock by prioritizing the execution of packet processing events. We classify simulation events into *high-priority events* that need strict timeliness like packet transmissions and *low-priority events* that are allowed

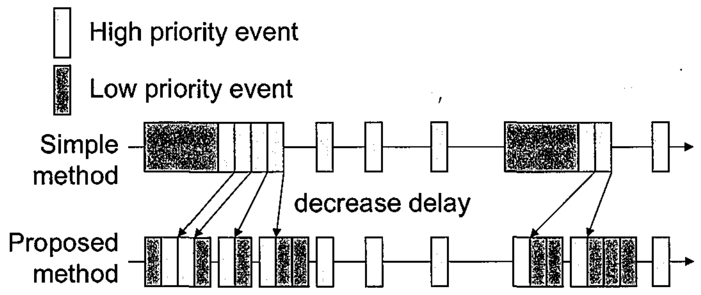


Figure 5.1: Overview of event processing method.

to be delayed for a while like mobility calculation, mobility trace processing, statistical data processing and so on. Then we fragment low-priority events into smaller processing units and execute each unit when there is no high-priority event to be processed.

First, we explain how real-time simulation progresses in TROWA. TROWA adopts Discrete Event Simulation (DES). In DES, simulation progresses by executing events in the order of their timestamps. All events are stored in the event queue in the ascending order of the timestamps. Then the event at the head is dequeued and executed and new events generated by this execution are queued.

In TROWA, we use the following simple scheme to let the simulator clock synchronize with real time. First, the simulator gets the real clock value T_0 when the simulation starts (the simulator clock is 0 at this moment). Then the simulator executes the next event as soon as the current value T_{now} of the real clock satisfies $T_{now} - T_0 \geq S_{next}$ where S_{next} is the timestamp of the next event.

As seen above, in real-time simulation each event should occur along the real clock and this differs from DES which processes each event right after the previous event. Therefore, in the proposed method, we use the waiting time between two events to execute low-priority events. We show the overview of the proposed event processing method in Fig. 5.1. In the proposed method, the simulator fragments each low-priority event into smaller processing units. Each processing unit should be small enough to be able to expect that its processing time is not greater than a certain threshold δ. Then the simulator executes each unit if residual waiting time is larger than δ. Even in this scheme, in order to avoid too large delay of low-priority events, we should set their appropriate deadlines and prioritize them when their deadlines are approaching.

To handle the PDU transmitted from real terminals, sockets of real terminals are polled after execution of each simulation event, and process PDUs immediately if there are received PDUs. In addition, if there is no event that can be processed, the simulator waits until the processing time of the next event by watching the sockets.

We think that my proposed method prioritizing simulation events can be applied to the existing parallel and distributed simulation techniques. It is important to consider that parallel simulation makes some processes (machines) wait for synchronization with other processes. This means that if some processes delay their low-priority events, it may make synchronization interval longer. Solving this problem in TROWA is part of my future work.

5.2 Experiment

In order to evaluate the proposed simulation method, we conducted some experiments to validate the number of nodes that can be simulated in real time and the accuracy (timeliness) of the event processing time.

In the experiments, we regarded mobility calculation events as low-priority events. We measured the delay from the real clock for each high- or low-priority event with and without the proposed method. Since my method sacrifices the preciseness of node positions to a certain degree by delaying position update events, we also measured the error of the nodes' positions to see the influence.

We used an ordinary PC (Pentium4 3.40GHz, 2GB memory) for executing the simulator and two laptop PCs (PentiumM 1.6GHz, 1.5GB memory) as real terminals for executing application programs. In the experiment, the one real application send 1024 bytes data at 0.777s interval to the other application. The fixed simulated nodes are placed to let the simulated communication path between the real nodes be 3 hops. In addition, we also placed simulated nodes which follow the trace mobility with the 1 sec. interval granularity (moving nodes). Those nodes send single-hop broadcast packets at the interval randomly chosen between 10 sec. and 20 sec. The transmission range was set to 100 meters and the duration of the scenario was 100 sec. We used the application layer API. The size of the field was 500m × 500m. The deadline of the mobility calculation

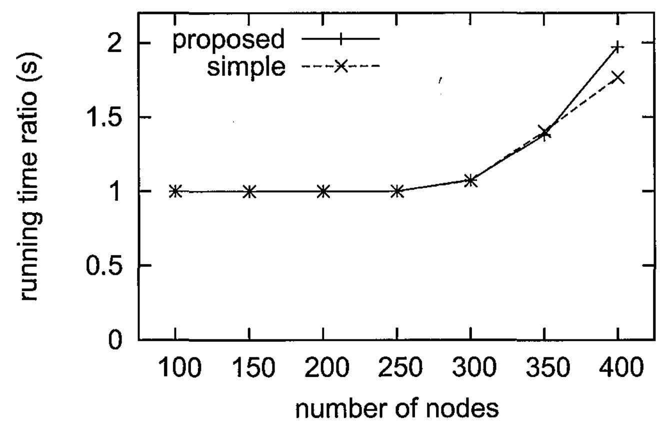


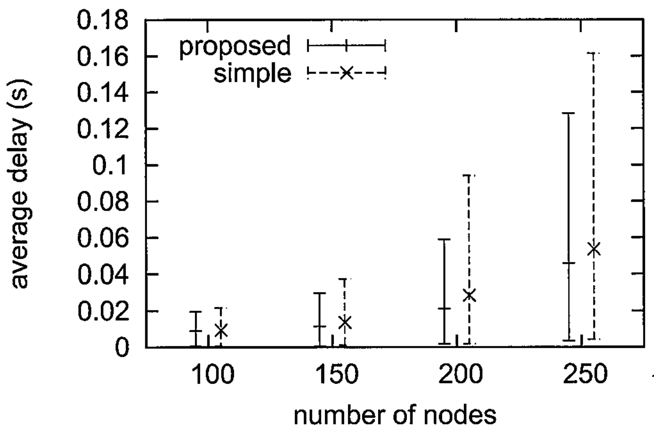
Figure 5.2: Running time of 100s simulation.

event was set to 1 sec., so each position update event was allowed to be delayed just before the next position update event.

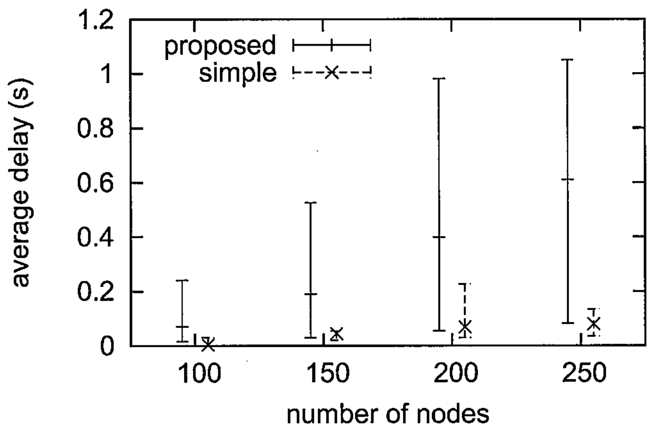
In Fig. 5.2, we show the ratio of the actual running time (i.e., time to complete simulation in real clock) to the specified simulation time changing the number of moving nodes from 100 to 400. In the figure, "proposed" and "simple" represent the proposed method which executes low-priority events during waiting time and "simple method" which executes all events in the order of their timestamps, respectively. Fig. 5.2 suggests that with this simulation scenario we can simulate up to 250 nodes in real time. We can see a little overhead of the proposed method for over 300 nodes.

Next, in Fig. 5.3, we show the average delay and confidence interval (95%) of the time when the events were actually processed from the real clock, changing the number of nodes. The figure shows that the proposed technique can reduce the average delay of the high-priority events by up to 25% by delaying low-priority events. For the 300 nodes case, the delay of the high-priority events of "proposed" and "simple" were 2.2 sec. and 1.7 sec., respectively. As a result, in my current implementation and given settings, networks with up to 250 nodes can be simulated in real-time without serious delay.

Fig. 5.4 shows the distribution of packet arrival interval to the application at the destination (real terminal). 'simulation' is the distribution of packet



(a) Delay of high-priority events



(b) Delay of low-priority events

Figure 5.3: Delay of event processing clock from real clock.

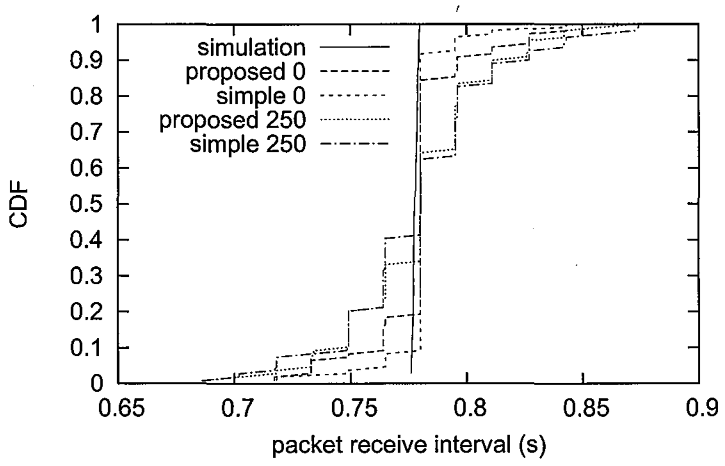


Figure 5.4: Distribution of packet arrival interval.

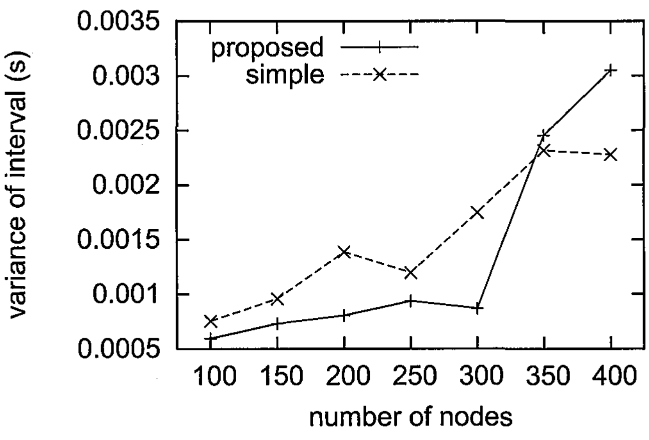


Figure 5.5: Variance of packet arrival interval.

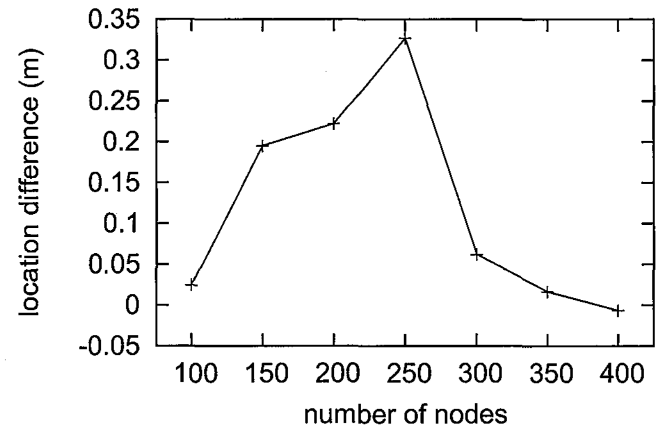


Figure 5.6: Average distance of the node location.

arrival interval in the simulation time when we use the sequential (not real-time) simulation and the application implemented in the simulator instead of real application. The number of moving nodes was set to 0 or 250. We assume that 'simulation' is the ideal distribution. The difference of 'proposed' and 'simple' from 'simulation' depends on the delay of high-priority events in Fig. 5.3. Also, Fig. 5.5 shows the variance of packet arrival interval of the above case. The ideal value ('simulation' in Fig. 5.4) is close to 0. We can see that my proposed method reduces the variance.

Next, we show the average distance of the moving nodes' positions between with and without the proposed prioritizing method in Fig. 5.6. The error was sampled just before the update of the node positions in the proposed method. We can say that the difference is really small compared with the transmission range (100m), so the impact of this difference on the accuracy of simulation results is expected to be quite small.

5.3 Conclusion

In this chapter, we explained the real-time simulation method used in the MobiREAL simulator and introduced the real-time simulation technique by prioritizing some simulation events. The MobiREAL simulator processes the network simulation events prior to the mobility simulation events, focusing on the difference of the time-scale between the network simulation events and the mobility simulation events. Of course, the proposed method can be applied to other separation policies of simulation events. The experimental results were shown and the average delay of the prioritized event is reduced by up to 25% while keeping the location error small enough.

Chapter 6

Conclusion

In this paper, we have introduced a realistic mobility model called Urban Pedestrian Flows for the wireless network simulation in the city section environment, and a wireless network simulator MobiREAL which incorporates a relational mobility description model called Condition Probability Event and run-time mobility operation function using real-time simulation.

By the UPF model, we can reproduce realistic pedestrian flows from data obtained by simple observation such as fixed-point observation using digital cameras or web cameras. To my best knowledge, no existing method considers this research direction. Also, through some experiments, we showed that we could design several MANET protocols and applications and verify their performance on the network topology, which was created by the UPF model and had never appeared in the existing random-based mobility models.

In addition, using the CPE model, we can enhance the reality of the microscopic mobility such as stopping at traffic signals and slowing down due to congestion. The relation between the network application and node mobility can also be described by this model, so the mobility and the network status may affect each other in the scenario produced by the CPE model. We have shown that dynamic behavior change reproduced by the CPE model affects the performance of the MANET system through the case-study. The discussion about the reality and the influence of this mobility still remains for future research, though we believe that my UPF/CPE framework provides reasonable fidelity of mobility modeling to conduct the performance evaluation of MANET systems in realistic environments.

Moreover, application programs on real mobile terminals can be tested and verified through the simulated network in the MobiREAL simulator. We can modify the movement of some specified mobile nodes and/or to give user inputs to the application programs running on those mobile terminals in real-time. Usually, we reproduce the specific situation for the worst-case test and the debug with an artificial and unrealistic movement of nodes or wait until accidentally occurs. However, the run-time operation function allows users to create and try various situations by removing implementation process that may need a large cost. For example, in order to validate the performance of route repair process in MANET routing protocols in realistic environment, it is very difficult to know in advance on which nodes the route is established due to dynamics of network topology and unpredicted movement of nodes in real environment. Actually, we have evaluated this situation in my experiment, as well as another experiment which tested user perception on VoIP quality over MANET.

My future work includes more detailed validation of the influence and fidelity of my mobility models, comparison of my mobility models and real traces, and the evaluation of several mobile ad-hoc communication protocols using the UPF/CPE mobility. The MobiREAL simulator supports parallelized DES [56]. We also plan to speed up event processing by combining the DES and proposed event prioritizing method for the real-time simulation.

Acknowledgement

I would like to gratefully acknowledge the enthusiastic supervision and great support of Professor Teruo Higashino, Osaka University, Japan during this research.

I am very grateful to Professor Makoto Imase, Professor Koso Murakami, Professor Masayuki Murata and Professor Hirotaka Nakano of Osaka University for their invaluable comments and helpful suggestions concerning this thesis.

I thank Professor Akio Nakata of Hiroshima City University for his precious advices and technical discussions provided through out the research.

In addition, I would like to thank Associate Professor Keiichi Yasumoto of Nara Institute of Science and Technology for his helpful advice and for many impressive comments.

And I am grateful very much to Associate Professor Hirozumi Yamaguchi for the precious supervision gave all over the research.

Also I would like to be thankful very much to Assistant Professor Takaaki Umedu and Assistant Professor Akihito Hiromori for the precious advices.

Also I heartily thank to my research partners, Mr. Kazuki Sato, Mr. Kazuki Konishi and Mr. Keisuke Nakata.

Finally, I would like to thank to everyone of Higashino laboratory for their feedback, encouragement and support.

Bibliography

[1] Tracy Camp, Jeff Boleng, and Vanessa 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*, 2(5):483–502, 2002.

[2] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. The IMPORTANT framework for analyzing the impact of mobility on performance of routing for ad hoc networks. *AdHoc Networks Journal*, 1(4):383–403, November 2003.

[3] Narayanan Sadagopan, Fan Bai, Bhaskar Krishnamachari, and Ahmed Helmy. PATHS: analysis of PATH duration statistics and their impact on reactive MANET routing protocols. In *Proc. ACM MobiHoc*, pages 245–256, 2003.

[4] Brent Ishibashi and Raouf Boutaba. Topology and mobility considerations in mobile ad hoc networks. *Ad Hoc Networks*, 3(6):762–776, 2005.

[5] Kumiko Maeda, Kazuki Sato, Kazuki Konishi, Akiko Yamasaki, Akira Uchiyama, Hirozumi Yamaguchi, Keiichi Yasumoto, and Teruo Higashino. Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation. In *Proc. 8th ACM/IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 151–158, 2005.

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

[7] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM/IEEE MobiCom*, pages 85–97, 1998.

[8] Christian 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.

[9] T. Chu and I. Nikolaidis. Node density and connectivity properties of the random waypoint model. *Computer Communications*, 27(10):914–922, 2004.

[10] Andres Rojas, Philip Branch, and Grenville Armitage. Experimental validation of the random waypoint mobility model through a real world mobility trace for large geographical areas. In *Proc. 8th ACM/IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 174–177, 2005.

[11] Esa Hyytia, Pasi Lassila, and Jorma Virtamo. Spatial node distribution of the random waypoint mobility model with applications. *IEEE Transactions on Mobile Computing*, 5(6):680–694, 2006.

[12] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *Proc. IEEE Infocom*, volume 2, pages 1312–1321, 2003.

[13] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Sound mobility models. In *Proc. ACM MobiCom*, pages 205–216, 2003.

[14] Jungkeun Yoon, Mingyan Liu, and Brian Noble. A general framework to construct stationary mobility models for the simulation of mobile networks. *IEEE Transactions on Mobile Computing*, 5(7):860–871, 2006.

[15] Guolong Lin, Guevara Noubir, and Rajmohan Rajaraman. Mobility models for ad hoc network simulation. In *Proc. IEEE Infocom*, volume 1, pages 454–463, 2004.

[16] William Navidi and Tracy Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, 03(1):99–108, 2004.

[17] Michael McGuire. Stationary distributions of random walk mobility models for wireless ad hoc networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 90–98, 2005.

[18] Philippe Nain, Don Towsley, Benyuan Liu, and Zhen Liu. Properties of random direction models. In *Proc. IEEE Infocom*, volume 3, pages 1897–1907, 2005.

[19] Amit Jardosh, Elizabeth M. BeldingRoyer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proc. ACM MobiCom*, pages 217–229, 2003.

[20] Matthias Hollick, Tronje Krop, Jens Schmitt, Hans-Peter Huth, and Ralf Steinmetz. Modeling mobility and workload for wireless metropolitan area networks. *Computer Communications*, 27(8):751–761, 2004.

[21] Wei-Jen Hsu, Kashyap Merchant, Haw-Wei Shu, Chih-Hsin Hsu, and Ahmed Helmy. Weighted waypoint mobility model and its impact on ad hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1):59–63, 2005.

[22] Amit Kumar Saha and David B. Johnson. Modeling mobility for vehicular ad-hoc networks. In *Proc. 1st ACM Workshop on Vehicular Ad Hoc Networks (VANET 2004)*, pages 91–92, 2004. (poster paper).

[23] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. An ad hoc mobility model founded on social network theory. In *Proc. ACM/IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 20–24, 2004.

[24] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *Proc. ACM/IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 53–60, 1999.

[25] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: A library for the parallel simulation of large-scale wireless networks. In *Proc. ACM Parallel and Distributeal Simulation (PADS'98)*, pages 154–161, 1998.

[26] *ns-2*. `http://www.isi.edu/nsnam/`.

[27] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. In *Proc. ACM MobiCom*, pages 107–118, 2002.

[28] Diane Tang and Mary Baker. Analysis of a metropolitan-area wireless network. *Wireless Networks*, 8(2/3):107–120, 2002.

[29] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless lan. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):195–205, 2002.

[30] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proc. ACM MobiSys*, pages 303–316, 2003.

[31] Suttipong Thajchayapong and Jon M. Peha. Mobility patterns in microcellular wireless networks. *IEEE Transactions on Mobile Computing*, 5(1):52–63, 2006.

[32] Marvin McNett and Geoffrey M. Voelker. Access and mobility of wireless pda users. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(2):40–55, 2005.

[33] Minkyong Kim, David Kotz, and Songkuk Kim. Extracting a mobility model from real user traces. In *Proc. IEEE Infocom*, April 2006.

[34] *lp_solve*. `http://groups.yahoo.com/group/lp_solve/`.

[35] Ian F. Akyildiz, Joseph S. M. Ho, and Yi-Bing Lin. Movement-based location update and selective paging for pcs networks. *IEEE/ACM Transactions on Networking*, 4(4):629–638, 1996.

[36] Gökhan Yavas, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data Knowledge Engineering*, 54(2):121–146, 2005.

[37] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: A survey. *Computer Networks*, 47(4):445–487, 2005.

[38] Tong Liu, Paramvir Bahl, and Imrich Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless networks. *IEEE Journal on Selected Areas in Communications*, 16(6):922–936, 1998.

[39] Wee-Seng Soh and Hyong S. Kim. Qos provisioning in cellular networks based on mobility prediction techniques. *IEEE Communications Magazine*, 41(1):86–92, 2003.

[40] Yun Won Chung, Dan Keun Sung, and A. Hamid Aghvami. Effect of uncertainty of the position of mobile terminals on the paging cost of an improved movement-based registration scheme. *IEICE Transactions on Communications*, E86-B(2):859–861, 2003.

[41] *OPNET*. `http://www.opnet.com/`.

[42] *QualNet*. `http://www.scalable-networks.com/`.

[43] Toshiyuki Miyachi, Ken ichi Chinen, and Yoichi Shinoda. StarBED and SpringOS: large-scale general purpose network testbed and supporting software. In *Proc. 1st international conference on Performance evaluation methodolgies and tools (valuetools '06)*, page 30, 2006.

[44] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC 2005)*, volume 3, pages 1664–1669, 2005.

[45] Daniel Mahrenholz and Svilen Ivanov. Real-time network emulation with ns-2. In *Proc. 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications(DS-RT '04)*, pages 29–36, 2004.

[46] Junlan Zhou, Zhengrong Ji, and Rajive Bagrodia. TWINE: A hybrid emulation testbed for wireless networks and applications. In *Proc. IEEE Infocom*, 2006.

[47] Erik Nordstrom, Per Gunningberg, and Henrik Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Proc. 1st Int. Conf. on Testbeds and Research Infrastructures for*

the DEvelopment of NeTworks and COMmunities (TRIDENTCOM 2005), pages 100–109, 2005.

[48] Priya Mahadevan, Adolfo Rodriguez, David Becker, and Amin Vahdat. MobiNet: a scalable emulation infrastructure for ad hoc and wireless networks. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 10(2):26–37, 2006.

[49] David Johnson, Tim Stack, Russ Fish, Daniel Montrallo Flickinger, Leigh Stoller, Robert Ricci, and Jay Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *Proc. IEEE Infocom*, 2006.

[50] Pradipta De, Ashish Raniwala, Srikant Sharma, and Tzi cker Chiueh. MiNT: A miniaturized network testbed for mobile wireless research. In *Proc. IEEE Infocom*, volume 4, pages 2731–2742, 2005.

[51] Yongguang Zhang and Wei Li. An integrated environment for testing mobile ad-hoc networks. In *Proc. ACM MobiHoc*, pages 104–111, 2002.

[52] Maneesh Varshney, Zhiguo Xu, Shrinivas Mohan, Yi Yang, Defeng Xu, and Rajive Bagrodia. WHYNET: a framework for in-situ evaluation of heterogeneous mobile wireless systems. In *Proc. of ACM WinTECH '07*, pages 35–42, 2007.

[53] Jean-Yves Le Boudec and Milan Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *Proc. IEEE Infocom*, volume 4, pages 2743–2754, 2005.

[54] *MobiREAL Simulator Web Page.* `http://www.mobireal.net/`.

[55] Shigeru Chiba and Muga Nishizawa. An easy-to-use toolkit for efficient java bytecode translators. In *Proc. 2nd Int. Conf. on Generative Programming and Component Engineering (GPCE '03)*, pages 364–376, 2003.

[56] Kazuki Konishi, Hirozumi Yamaguchi, and Teruo Higashino. Efficient parallel simulation of mobile wireless networks by run-time prediction of multihop propagation delay. In *Proc. 3rd IEEE Int. Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007)*, 2007.