Cooperative Network Monitoring and Ad-hoc Topology Formation for Increasing Network Reliability

January 2008

Akira UCHIYAMA

Cooperative Network Monitoring and Ad-hoc Topology Formation for Increasing Network Reliability

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2008

Akira UCHIYAMA

List of Publications

Journal Papers

1. Akira Uchiyama, Takaaki Umedu, Keiichi Yasumoto and Teruo Higashino: "A Proposal of Distributed Network Monitor with Autonomous Group Formation", *IPSJ Journal*, Vol.47, No.2, pp.446–454 (February 2006) (in Japanese).

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

3. Sae Fujii, Akira Uchiyama, Kumiko Maeda, Takaaki Umedu, Hirozumi Yamaguchi and Teruo Higashino: "Proposal and Evaluation of Localization Algorithm for Mobile Nodes and Sparsely Deployed Landmarks", *IPSJ Journal*, Vol.48, No.12, pp.3977–3986 (December 2007) (in Japanese).

Conference Papers

1. Akira Uchiyama, Takaaki Umedu, Keiichi Yasumoto and Teruo Higashino: "FLEXA: Distributed and Flexible Network Monitoring with Autonomous Group Formation", *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM2005)*, No.1, pp.2305–2309, Nice, France (May 2005).

2. Akira Uchiyama, Takaaki Umedu, Keiichi Yasumoto and Teruo Higashino: "Efficient and Robust Distributed Network Monitoring using Dynamic

Group Formation", *Proceedings of the 10th IFIP/IEEE Network Operations & Management Symposium (NOMS2006)*, Vol.10, No.1, pp.900–903, Vancouver, Canada (April 2006).

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) Mini-Symposium*, Vol.26, No.1, pp.2306–2310, Alaska, USA (May 2007).

Other Related Conference Papers

1. 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, Canada (October 2005).

2. Takurou Sakai, Akira Uchiyama, Yoshitaka Nakamura and Teruo Higashino: "Certification of Encounter History Among Low Power Mobile Sensors", *Proceedings of the 4th Annual IFIP WG 11.9 International Conference on Digital Forensics (ICDF2008)*, Kyoto, Japan (January 2008) (in press).

3. Sae Fujii, Akira Uchiyama, Takaaki Umedu, Hirozumi Yamaguchi and Teruo Higashino: "An Off-line Algorithm to Estimate Trajectories of Mobile Nodes Using Ad-hoc Communication", *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom2008)*, Hong Kong, China (March 2008) (in press).

4. Hiroaki Urabe, Masatoshi Nakamura, Akira Uchiyama, Takaaki Umedu and Teruo Higashino: "Realistic Mobility Aware Information Gathering in Disaster Areas" *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC2008)*, Las Vegas, USA (March 2008) (in press).

Abstract

With the advent of the Internet, communication networks are now a fundamental social infrastructure. Also, hand-held terminals and laptop computers become widely used and they will form mobile ad-hoc networks (MANETs), which are expected to realize various applications and services in the future. Since many networked applications and services require secure and stable communication, we need to increase stability and reliability of those networks.

In wired networks, as the number of managed network segments increases, workloads of network administrators become heavy because they must handle a lot of management data obtained from those network segments. Moreover, we face threats such as DDoS attacks caused by intelligent bots, which are difficult to detect unlike conventional viruses. In such a situation, an efficient network monitoring technique is required to enhance reliability of wired networks. Meanwhile, performance of applications/protocols on MANETs varies from minute to minute due to the movement of nodes. For example, node mobility brings dynamics of topology and density, which result in link disconnection and/or packet collision. For this reason, it is significant to consider mobility patterns of nodes and involved topology dynamics in order to increase reliability of MANETs.

Considering the above facts, this thesis studies the following two research topics: (1) a network monitoring technique which reduces workloads of network administrators and achieves high availability for the administrator to receive the monitored information reliably and timely, and (2) ad-hoc topology formation which supports stable communication on MANETs.

The first method aims at reducing workloads of network administrators and detecting damaged areas quickly and thoroughly when security problems occur. For this purpose, we propose a middleware for efficient and robust network monitoring based on autonomous group formation. In this technique, we specify a

management scenario, which is composed of functions and primitives provided by our middleware. The proposed middleware provides several APIs which support group formation, communication among group members and gathering information about monitoring items such as average TCP traffic. The administrators can easily describe management scenarios by using the provided APIs. Due to the dynamic group formation function, members in the same group can share information and send the summarized information to the administrator. The proposed technique allows any member of the group to send the shared information to the administrator node. Therefore, whenever link/node failures occur, the administrator can receive information if at least one group member succeeds in sending the information.

The second method aims at realizing stable communication on MANETs in urban areas, and proposes an ad-hoc topology formation technique considering movements of nodes. We have modeled realistic behavior in urban areas which has not been considered in the random-based mobility models, and evaluated performance of protocols and applications on MANETs through simulation using the mobility model. In order to confirm the importance of mobility consideration on designing applications and protocols in urban areas, we have compared the performance of the routing protocol DSR with its modified version. In the modified DSR, multi-hop routes are established using communication links which are expected to be long-lived due to similarity of mobility directions of nodes. Simulation results have shown that the modified DSR could improve route stability by considering mobility patterns.

For more reliability of MANETs, we may use more information about context and environment, and location information is the most important one among them. To make use of location information more easily, we present a range-free ad-hoc localization algorithm called UPL (Urban Pedestrians Localization), for positioning mobile nodes in urban districts. The design principle of UPL is two-fold. (1) We assume that location seeds are deployed sparsely due to deployment-cost constraints. Thus most mobile nodes cannot expect to meet these location seeds frequently. Therefore, each mobile node in UPL relies on location information received from its neighboring mobile nodes in order to estimate its area of presence. The area of presence of each mobile node becomes inexact as it moves, but it is helpful to reduce the areas of presence of the

other mobile nodes. (2) To predict the areas of presence of mobile nodes accurately under mobility, we employ information about obstacles such as walls, and present an algorithm to calculate the movable areas of mobile nodes considering obstacles. This also helps to reduce each node's area of presence. The experimental results have shown that UPL could achieve 8m of position errors in a real region of Osaka downtown when the wireless communication range was 10m. In addition, as an application example, we have applied UPL to the geographic routing protocol GPSR which utilizes locally-constructed planar graphs based on location information to choose the next hop. Simulation results have shown that UPL could help GPSR to be GPS-free, without excessively sacrificing its performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advent of the Internet, communication networks are now a fundamental social infrastructure. Also, hand-held terminals and laptop computers become widely used and they form mobile ad-hoc networks (MANETs) [1, 2], which are expected to realize various applications and services in the future. The future trends of mobile wireless ad-hoc networks are presented in [3, 4]. For example, if disasters disable cellular networks, ad-hoc networks will be composed of cellular phones with wireless LAN devices and used to collect/distribute safety information from people in the disaster area. Thus MANETs will be one of key infrastructures to make our life safer and more affluent. Since many networked applications and services require secure and stable communication, we need to increase stability and reliability of those networks.

In wired networks, as the number of managed network segments increases, workloads of network administrators become heavy because they must handle a lot of management data obtained from those network segments. Moreover, we face threats such as viruses and DoS/DDoS attacks [5, 6, 7]. In such a situation, an efficient network monitoring technique is required to enhance reliability of wired networks. Meanwhile, performance of applications/protocols on MANETs varies from minute to minute due to the movement of nodes. For example, node mobility brings dynamics of topology and density, which result in link disconnection and/or packet collision. For this reason, it is significant to consider mobility patterns of nodes and involved topology dynamics in order to increase reliability of MANETs.

Considering the above facts, this thesis studies the following two research

topics: (1) a network monitoring technique which reduces workloads of network administrators and achieves high availability for the administrator to receive the monitored information thoroughly and timely, and (2) ad-hoc topology formation which supports stable communication on MANETs.

The first method aims at reducing workloads of network administrators and detecting damaged areas quickly and thoroughly when security problems such as infections of viruses and DoS/DDoS attacks occur. In recent years, damage caused by viruses and DDoS attacks has remarkably been increasing. To make matters worse, there are concerns about DDoS attacks and a new type of spam on voice networks exploiting bot nets [8], which are intelligent and difficult to detect unlike conventional viruses. Bot nets are overlay networks composed of networked computers where malicious softwares called bots are installed. The owners of bot nets can send commands to the bots through IRC networks and easily send spams, launch DDoS attacks and even update bots. In addition, since bots consume little CPU power and network bandwidth, it is difficult to detect. There may exist a lot of bots and most computer users do not notice the facts. Thus potential threats of DDoS attacks may be growing.

This fact suggests the necessity of efficient network monitoring and management techniques which allow administrators to quickly notice failures or abnormal situations in managed networks. In wide area network monitoring, attacks, intrusions or infections are likely to occur on multiple network segments at the same time. So, it is desirable for the network monitoring system to have a certain level of autonomy which allows intelligent monitoring without interactions with the centralized administrator. Also, malicious attacks would often cause node failures and network congestion. These situations prevent the administrator from collecting management information and identifying the damaged area. So, the information availability, which means timely and reliable delivery of monitored information to the administrator, is an important criterion to assess the capability of the monitoring system. Moreover, as the damaged area becomes large (i.e. the number of damaged segments increases), the total amount of the monitored information will also become large. Therefore, the monitoring system should be scalable in terms of the number of managed network segments. Here, avoiding traffic convergence to the administrator node is considered important for scalability.

For this purpose, we propose a framework for efficient and robust distributed network monitoring and develop a middleware based on the proposed technique. In our technique, multiple agents are installed on the corresponding managed network segments. For the installed agents, we can specify a management scenario which describes the reactive behavior to the problems that are likely to occur. Then, they execute the given management scenario. Based on the scenario, those agents gather information on their responsible segments, and when they detect problems, (i) they autonomously form groups, (ii) they share the information gathered by all agents in the group by exchanging the information among them, and (iii) they autonomously solve the problems without interaction with the administrator node. The administrator node can also receive information collected by those groups and give them additional instructions based on the information and previous experiences. If the number of the group members becomes large, the communication cost in the group becomes large and the communication itself may be unreliable during congestion. In the proposed technique, for efficiency and robustness, the network segments detecting the same problem form multiple groups with appropriately limited group size, and a hierarchical structure among those groups is autonomously constructed. In order to improve availability, the proposed API for group communication provides information sharing among the group members. Also, in our management scenarios, any group member is allowed to send the shared information to the administrator so as to guarantee the reliability of information delivery.

Additionally, in order for timely and guaranteed detection of problems, the administrator can describe a management scenario so that group formation is carried out among physically-separated network segments. In general network monitoring, neighborhood relations among network segments are basically given by *physical* link connection, and a hierarchical tree structure is constructed for reporting the monitoring information from those network segments to the administrator node. However, the area damaged by brand-new viruses and DDoS attacks is often spread over physically-separated network segments. For example, the same company's servers located in different network segments may be attacked simultaneously. In such a case, from the experiments of previous DDoS attacks, the network administrators may be able to estimate critical network segments that should be monitored cooperatively. Therefore, grasping such logical

relations between network segments and monitoring those segments in parallel are very important to detect DDoS attacks timely and thoroughly. From experimental results, we show the effectiveness of *logical* neighborhood relations for quick and accurate detection of problems occurring in multiple segments.

Next, the second method aims at realizing stable communication on MANETs in urban areas, and proposes an ad-hoc topology formation technique considering movements of nodes. In urban areas, there exists several main flows of pedestrians since most pedestrians walk along main streets such as pedestrianized zones and follow the shortest-paths between major spots. However, we usually use simple mobility models such as the Random Way Point (RWP) model [9] without any obstacles, for simulating MANET protocols. This is because such simple mobility models are commonly available in many simulators and also they can be used to compare the performance of different systems and protocols under the same mobility settings. Moreover, several variants of RWP are proposed in Refs. [10, 11]. On the other hand, Ref. [12] presented the impact of mobility to the performance of routing protocols on MANETs and some others have also dealt with this topic [13, 14, 15]. The results presented in those literatures encourage us to address the important fact that performance of protocols expected in real environment cannot be measured through simulation with random-based mobility models such as Random Way Point. We have learned a lot of similar insights from these literatures, however, these literatures basically focus on specific protocols (mainly routing protocols) and thus do not deal with new emerging applications for future ubiquitous society. In addition, they do not provide methodologies to evaluate the performance of such services in real environment. Thus, we have analyzed the effect of mobility patterns on such new emerging applications on MANETs.

The analysis shows that communication links between neighbors are more stable in the realistic model than a random-based model because the realistic model captures the characteristics of pedestrians where many of them go into the same direction along the streets and their relative speed is low in urban areas. This fact implies the importance of mobility consideration on designing applications and protocols in urban areas. To prove this, we have compared the routing protocol DSR with its modified version so that multi-hop routes are established on communication links which are expected to be long-lived due to

similarity of mobility directions of nodes. Simulation results have shown that the modified DSR can increase route stability by considering mobility patterns.

Thirdly, for more reliability of MANETs, we may use more information about context and environment, and location information is the most important one among them. To make use of location information more easily, we propose a range-free ad-hoc localization algorithm called UPL (Urban Pedestrians Localization), for positioning mobile nodes in urban districts. Location information is significant for future ubiquitous systems that provide with people in cities highly-personalized and reliable services such as route navigation, location-dependent advertisements and localized communication. For determining positions of devices, many localization algorithms have been presented so far.

Usually, localization algorithms assume that some location seeds advertise their accurate positions to nodes in their transmission ranges. Some of them assume a few number of seeds with longer transmission ranges (GPS falls into this category) and some others assume a large number of seeds with short transmission ranges in order to cover the target fields by these seeds. However, receiving signals from seeds over a distance requires clear lines of sights, which are sometimes hard to obtain between buildings, in underground cities and so on. Additionally, it is highly expensive to widely deploy a number of short-range location seeds. Another alternative is to assume mobile seeds to enhance the coverage of regions [16]. This works fine if these mobile seeds well cover the target field. However, depending on the size of the target field and the speeds and density of mobile seeds (especially, if we assume slow pedestrians in large cities), it is also hard for ordinary (non-seed) nodes to find the mobile seeds.

Some techniques exploit indirect information from seeds. Techniques categorized into "collaborative multi-lateration" assume that position information of seeds is delivered in a multi-hop way, and the distance to the seeds is approximated by additional information such as the number of hops on the wireless ad-hoc networks [17, 18, 19, 20, 21]. The other techniques categorized into "iterative multi-lateration" use the estimated position of a node to estimate positions of its neighbors iteratively [22, 23, 24]. However, these techniques may not work well if *the wireless ad-hoc networks are mobile and frequently partitioned*, which is true in urban cities. For example, pedestrians and vehicles in cities are not

stationary, and they are not always connected due to obstacles such as buildings and due to nonuniform deployment and density.

To overcome this problem, in this thesis, we present UPL designed for positioning mobile terminals in urban city areas. The key idea of UPL is following. As discussed earlier, in urban cities we do *not* assume that mobile nodes hear signals from location seeds frequently and also mobile nodes are fully-connected. Thus each mobile node in UPL maintains its own area of presence, and updates the area whenever it encounters other nodes and receives the information about the areas of presence of those nodes. Localization is performed by intersecting the two (or more) areas considering radio range. This "opportunistic" localization does not require well-connected ad-hoc networks. On the other hand, due to movement of mobile nodes, the area of presence expands as time passes. Let us consider an example in free space. At time T, we assume that a node's area of presence is represented as a circle of radius u centered at point c. If the maximum speed of the node is V, then the area of presence of the node at time $T + \triangle T$ can be estimated as the circle of radius $u + V \cdot \triangle T$ centered at c. This means that the area of presence expands quadratically with respect to V at each time unit. However, actually in urban cities mobile nodes do not move in free space but in space restricted by walls and streets. Considering this fact, expanding the area of presence in such a way is overcautious. Thus our algorithm fully utilizes obstacle information, and precisely determines the movable areas of mobile nodes considering the obstacles. To do so, each node only needs to know an obstacle map of its neighborhood. Unlike car navigation systems, we assume obstacle maps are simple and lightweight enough, and thus easy to distribute. We later discuss how obstacle maps can be distributed among mobile nodes.

To confirm the effectiveness of the algorithm, we have conducted realistic simulations using our open source mobile network simulator MobiREAL [25, 26, 27]. From the experimental results, UPL could achieve smaller position errors (8m in average) in a real region of Osaka downtown than two known localization methods. In addition, as an application example, we have applied UPL to the geographic routing protocol GPSR which utilizes locally-constructed planar graphs based on location information to choose the next hop. Simulation results have shown that UPL could help GPSR to be GPS-free, without excessively

sacrificing its performance.

The rest of this thesis is organized as follows. The next chapter reviews the related work. Chapter 3 details our cooperative network monitoring technique. Chapter 4 analyzes the mobility effect on the performance of MANET applications. Chapter 5 describes UPL, a range-free ad-hoc localization algorithm for positioning mobile nodes in urban districts. Finally, Chapter 6 summarizes this thesis.

Chapter 2

Related work

2.1 Network Monitoring Techniques

Several centralized monitoring tools have been proposed [28, 29]. In such techniques, management data from monitoring segments are concentrated to the centralized administrator when monitoring large-scale networks. Ref. [30] proposes a hybrid technique with both the polling and reactive monitoring techniques. In this technique, monitoring information is sent to the administrator node less frequently from monitoring segments if the monitored information from those segments has not changed since the previous report was sent to the administrator node. The main goal of those techniques is to collect monitoring information periodically so that the administrator node can recognize the entire traffic flow. On the other hand, the goal of our technique is early detection of unusual situations caused by DDoS or some other attacks.

Ref. [31] proposes a technique to make the monitoring system scalable by utilizing a hierarchical tree of nodes as the delivery paths for the gathered information. This technique, however, would need some time to identify areas with problems since the tree structure is static.

Distributed monitoring techniques using mobile agents have also been proposed [32, 33, 34, 35, 36]. These techniques let mobile agents travel around multiple managed network segments to gather information. This approach can significantly reduce the traffic around the administrator node by cooperative information gathering among managed segments. However, it may take some time for mobile agents to return to the administrator node, and sometimes they

may not be able to return timely due to network congestion.

Several group-based monitoring techniques have also been proposed. The three-tier architecture is adopted in HAMSA [37] where mid-term agents can dynamically be assigned to improve availability of the monitoring system. However, the administrator node still needs analysis of the received information to identify the area with problems. In general, the area where problems occur is difficult to predict beforehand. To cope with the cases beyond expectations, it is desirable to change members of each group flexibly to reduce communication overhead and to improve information availability. Although a dynamic group management technique is proposed in HiFi [38], its group communication is based on IP multicast and a reliable multicasting server is required. If the size of a group becomes very large and communication links among its group members are congested, some device is needed to realize reliable multicasting.

Additionally, several papers [39, 40, 41, 42] propose Intrusion Detection Systems (IDSs) which aim to detect viruses, malicious attacks and so on. Snort [43] is one of the most famous IDS tool. By using IDSs, network administrators can receive alert messages when specified cases (e.g. worm infection, port scan and so on.) occur. Our technique can co-work with these IDSs.

DoS/DDoS detection is also very important for network security. Ref. [44] proposes a simple and robust mechanism for detecting SYN flooding attacks. In this approach, leaf routers are points for detecting SYN flooding attacks. Our technique can be used to easily realize such a system in cooperation with IDSs.

2.2 Modeling of Node Mobility and Environment

In the stage of performance evaluation, we usually conduct computer simulation. For such a purpose, several mobility models have been presented. The random mobility model and Random Way Point (RWP) model [9] are most common models. Especially, the RWP model is well used where each node sets a new destination randomly after spending a pause time at the current destination. It is more realistic than the random mobility model where each node continues to randomly choose its new direction periodically. It is also simple to generate (we just need to set minimum and maximum speeds and a maximum pause

time) and useful to compare the performance of different protocols in the same condition. The properties of the RWP model have been studied for several years. For example, Ref. [10] presents a variant where nodes can change their directions smoothly, keeping analytical properties. Ref. [11] presents "sound mobility model" based on the RWP model, which is designed to keep the average velocities of nodes to exclude harmful effects to simulation results. Refs. [45, 46] provide analysis of the spatial node distribution in random-based models.

In addition to these analytical mobility models, realistic scenarios are also important to deploy applications and services in real environment. For example, if we know where the target service is used, we should create a set of scenarios that well-reproduce and well-characterize the mobility of nodes in that place. Ref. [47] introduces obstacles (i.e., buildings) in a given simulation field. Any polygon can be placed in the space and a realistic pathway between any two obstacles can automatically be generated using a Voronoi graph computation algorithm. The research group also provides plug-ins for GloMoSim and ns-2 simulators for the utilization of the proposed mobility. Ref. [48] proposes a macroscopic mobility model for wireless metropolitan area networks, where a simulation space 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. Ref. [49] presents 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 and transition probabilities of nodes for each pair of regions, it uses a Markov model in order to model node mobility between these regions.

Using real traces of mobile users is also helpful to reproduce real mobility [50, 51, 52, 53]. In the work of this category, user behavior patterns were analyzed by collecting real traces of users. Ref. [53] has collected traces from the wireless network at the ACM SIGCOMM2001 conference place, and given several analytic results. Similarly, Ref. [52] has collected traces from wireless network users in a metropolitan area wireless network. Also, Refs. [50, 51] give trace results and analysis in a campus-wide wireless network. Those approaches can simulate the detailed real world. The disadvantage of these approaches is that they require a very large amount of information collection to produce a mobility scenario.

For evaluation of MANET applications in realistic environment, we use our MobiREAL simulator. We can reproduce and model realistic environment by MobiREAL since MobiREAL enables us to describe behavior of nodes in detail. Overview of MobiREAL is described in Section 4.2.

2.3 Techniques for Stable Ad-hoc Communication

In order to achieve stable communication on MANETs, various techniques have been proposed so far. Ref. [54] presents a multi-path routing protocol which utilizes several paths between a source and its destination. For vehicular networks, GVGrid [55] tries to keep the initial forms of routes which are built using digital maps and location information.

Some routing protocols for wireless networks also consider stability of communication links and/or multihop routes. Ref. [56] proposes a Mobility-centric Data Dissemination algorithm intended for Vehicular networks (MDDV) which uses digital maps and GPS information. Messages are geographically forwarded along a predefined trajectory. MDDV determines more stable trajectories (i.e., multihop routes) by static information such as the number of lanes. In our techniques, we assume pedestrian nodes, so neither digital maps nor GPS is available. Several routing protocols that use the link metrics have been also presented. ABR (Associativity-Based Routing) [57] and SSA (Signal Stability based Adaptive routing) [58] measure connectivity relationship and signal strength respectively. PLBR (Preferred Link Based Routing) [59] can also exploit link metrics including stability. Different from the above two protocols, PLBR adopts a neighbor selection technique in forwarding RREQ messages used in DSR [60] where only preferred neighbors are allowed to forward messages in order to avoid the broadcast storm. The collected values are used to determine the best route.

Location information of nodes is also useful to improve reliability of MANETs communication. Assuming that each node can obtain its position (e.g. through GPS), many literatures have presented position-based routing methods (see Refs. [61, 62, 63] for surveys).

Several researches have presented position-based unicast, multicast and any-

cast routing protocols assuming that the location information of a specified destination node can be obtained by some location service. DREAM [64] proposes an unicast protocol where an efficient update method of position information of destination nodes using their distances and mobility rates is presented. GPSR [65] also presents a unicast routing protocol but can detour obstacles by using perimeter (or face) routing. The protocol in [66] uses a Voronoi diagram in combination with the mobility prediction of the destination node to determine neighboring nodes to which packets are forwarded. The protocol in [67] considers position-based multi-path routing for robustness, quality of service and other reasons. The above protocols localize the decision procedures to decide neighbors to which packets are forwarded in order to make them scale to the growth of the number of mobile nodes. Meanwhile, for small group multicast on MANET, DSM [68] gives an efficient and centralized technique for constructing multicast trees assuming that each node knows the geographic locations of all the other nodes.

Geocasting is also a form of position-based routing where destination nodes are those in a specified region (i.e. a sender specifies a region as a destination). Ref. [69] presents Location Based Multicast (LBM), a geocast protocol enhanced from Location Aided Routing (LAR) [70]. In LBM, each node forwards a message only to nodes within a certain area (the smallest rectangle which contains both the sender and the destination region) called a forwarding zone. The forwarding zone can be extended or shrunk by a parameter δ, depending on obstacles, mobile node density and so on. GeoGRID [71] partitions a geographic space into squares (grids). In each grid, a node called a gateway is selected and is responsible for forwarding packets to neighboring grids. There are two different types of message forwarding policies, flood-based and ticket-based. The former one is similar with LBM, while the latter one can control the number of messages by tickets, which are issued by a source and mean the permission to send messages. OFSGP/OFMGP [72] also partitions a geographic space into cells (hexagons). Direction-based depth-first search is performed to detour obstacles. Geomulticast [73] also presents a cell-based method and adopts a distance-based greedy forwarding policy. GAMER [74] is a multipath routing based on MGRP [75]. This adopts a similar forwarding policy as LBM but differs in maintaining and adapting the created mesh topology. From

the viewpoint of forwarding policies, Refs. [64, 66, 67, 69, 71, 73, 74] adopt greedy forwarding which may be a simpler form, while Refs. [65, 72, 76] adopt a hybrid form of greedy forwarding and face routing (messages traverse along obstacles [77]) which will achieve higher route discovery ratios under existence of obstacles but usually require a larger number of messages.

2.4 Localization Techniques

Many efforts have been dedicated so far to investigate range-based positioning techniques, which rely on range measurement techniques such as RSSI, AOA and TOA. For example, Refs. [78, 79] investigate to improve the quality of RSSI-based measurement. Ref. [80] presents an implementation of AOA system on IEEE802.11. Design and implementation of several location support systems are presented such as RADAR [78], a cricket location-support system [81] that supports indoor positioning, and its application to object tracking [82].

On the contrary, *range-free* techniques are cost effective alternatives, since they require no additional infrastructure except location seeds and communication function between nodes. Thus it is well-fit for ad-hoc wireless networks.

Ref. [83] has presented the following categories of localization algorithms; hyperbolic tri-lateration (positioning by distances from three landmarks), triangulation (if angles are obtained via Angle of Arrival (AOA)) and maximum likelihood multi-lateration (taking the most likelihood region to estimate the position). Multi-lateration is further classified into the following three simple categories, called *atomic*, *collaborative* and *iterative* multi-lateration. The atomic multi-lateration uses more than two seeds to determine a position. In the collaborative multi-lateration, nodes exchange landmark information cooperatively to be used for estimation. The iterative multi-lateration uses estimated positions as new reference points and estimates the other positions iteratively. Since our proposed UPL exploits range-free multi-lateration, we survey the related multi-lateration techniques.

First, as the atomic multi-lateration techniques, Centroid [84] assumes seeds with long transmission radius or a large number of seeds so as to cover the target fields. Each node estimates its position by calculating centroid of all the seeds it hears. APIT [85] uses a set of triangles formed by seeds which have long transmission radius, and determines the location by checking that a node is inside or

outside of each triangle. MCL [16] takes a different approach where each node updates the set of its likelihood points whenever it hears a signal from a seed. Secondly, as the cooperative multi-lateration techniques, APS (Ad-hoc Positioning System) [17, 18] approximates the distance from an indirect seed by several ways, by estimation of hop distance or the number of hops. Ref. [19] presents a similar hop-based approach. Amorphous [20] is also a range-free localization that utilizes hop counts from seeds as well, where each node estimates its coordinates by finding coordinates that minimize the total squared error between the calculated and estimated distances. Self-configurable positioning [21] takes a similar approach where a distance is approximated via hop-based analysis, but then landmarks exchange the information to determine their coordinates followed by local coordination determination performed by each node. Ref. [86] presents MDS-MAP, which uses shortest path information between nodes to estimate the distance, and uses "multidimensional scaling" to determine the positions. Ref. [87] presents a method to map "proximity" information (such as hop count information) between nodes into geographic distance considering the characteristics of anisotropic networks. Most recently, Ref. [88] presents an algorithm for sensor positioning in concave areas. Finally, as the iterative multi-lateration, recursive position estimation in Ref. [22] presents a fundamental technique for iterative multi-lateration. Ref. [23] presents a theoretical limit of connectivity-based multi-hop positioning and provides an algorithm that outperforms hop-based algorithms. Ref. [24] presents a range-free algorithm called Sextant, which uses connectivity information. It represents a region of presence by Bezier curves to pursue the accuracy and constrain the region by interactive multi-lateration.

These methods will work fine in stationary networks, but we think that we may need a new design for *positioning mobile nodes in urban districts*. Most existing proposals rely on static ad-hoc networks because their target applications are mainly positioning of wireless sensor nodes. Different from these approaches, each node in UPL relies on indirect seed information, in particular, it relies on opportunistic encounter with the other mobile nodes. Since we encounter many people in cities and since hand-held devices including smart phones are now being equipped with personal area communication devices, this is a reasonable solution. Each node uses the area of presence of another node

with which it encounters, considering the movement of the node during the elapsed time since the last time when the area of presence was updated. Additionally, we use obstacle information to consider the movement of those nodes. We present an algorithm to compute the movement in a simple way such that it can be implemented in small devices.

Considering the above discussion, our contribution is that we present a new concept of localization which well fits positioning mobile nodes in urban districts, and propose an efficient localization algorithm which is simple enough. As far as we know, this is the only approach which deals with range-free ad-hoc localization in urban districts.

Chapter 3

Cooperative Network Monitoring using Autonomous Group Formation

3.1 Introduction

In this chapter, we propose a framework for efficient and robust distributed network monitoring and develop a middleware based on the proposed technique.

We have designed and implemented the middleware consisting of several APIs which facilitate easy description of monitoring scenarios. Through experiments using ns-2, we show that the proposed technique achieves the quick detection of problematic areas with reasonably small control traffic as well as it achieves higher availability than the existing static tree based monitoring techniques.

The rest of this chapter is organized as follows. Section 3.2 presents the framework of the proposed technique and Section 3.3 describes its implementation. Section 3.4 shows experiment results through ns-2 simulation. Finally, this chapter is concluded in Section 3.5.

3.2 Framework

3.2.1 Outline

In our technique, we use two types of agents: **sensor agents** and **manager agents**. Sensor agents are installed on the corresponding managed network segments and they monitor traffic, specific packets, or failures of network devices on their responsible network segments depending on the specified monitoring items. Hereafter, we call the information which each sensor agent has collected during certain time period, as the **management data**. On each segment, management data is gathered with SNMP, tcpdump, IDS such as Snort and so on. The amount of traffic with specific destination/source IP addresses or TCP/UDP ports can also be measured. Each sensor agent holds the addresses of its **neighbor agents** that can be placed on any segments[1]. When each sensor agent detects a problem, it tries to form a group with its neighbor agents which have detected the same problem. The group can autonomously expand until reaching the specified size. The agents which formed a group can exchange the information gathered on their segments and generate a digest (e.g., statistics) by summarizing the exchanged information. According to the shared information, they can autonomously narrow the monitoring items to identify the cause of the problem, and solve the problem, for example, by configuring packet filtering on network devices. The detailed behavior of sensor agents such as monitoring items and conditions to form groups with other agents is specified in a **management scenario**. Hereafter, we call sensor agents simply as agents.

The manager agent is executed on the administrator node and it distributes the management scenario to sensor agents, displays gathered information in real time to the administrator and so on. Also the administrator can change monitoring items or group formation conditions in the management scenario, and distribute the new scenario to specific sensor agents directly through the management agents.

3.2.2 Neighborhood Relations

Like the existing static tree based network monitoring, basically neighbor segments are defined based on physical neighborhood relations among network seg-

[1]The neighbor segments mean both physical and logical neighborhood relations. For details, see Section 3.2.2

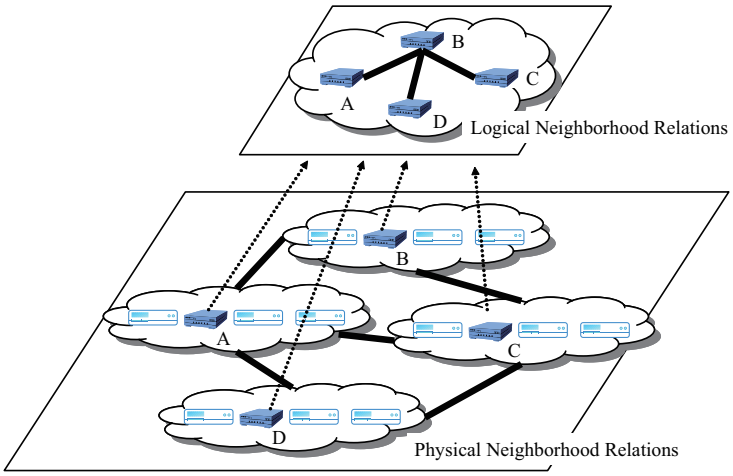


Figure 3.1: Physical/logical neighborhood relations.

ments. However, recently separate network segments are often attacked at the same time. Therefore, it is important to hold neighborhood relations among physically separate network segments. Here, such neighborhood relations are called logical neighborhood relations. In our technique, both the manager agents and sensor agents can specify the following logical neighborhood relations (see Fig.3.1).

- *between an intermediate node in the static tree and its ancestor node :* It is useful to inform information about a descendant's network segment to its ancestor node quickly when the descendant segment is attacked.

- *among network segments of the head office of a company and its branch :* If a company has several branches and their network segments are spread in physically separate network segments, multiple servers in the company may be attacked in parallel. This logical neighborhood relation is useful for detecting such attacks.

- *among previously attacked network segments :* In general, the security level of previously attacked network segments may not be so high. It is important to observe previously attacked network segments simultaneously in order to detect network attacks quickly.

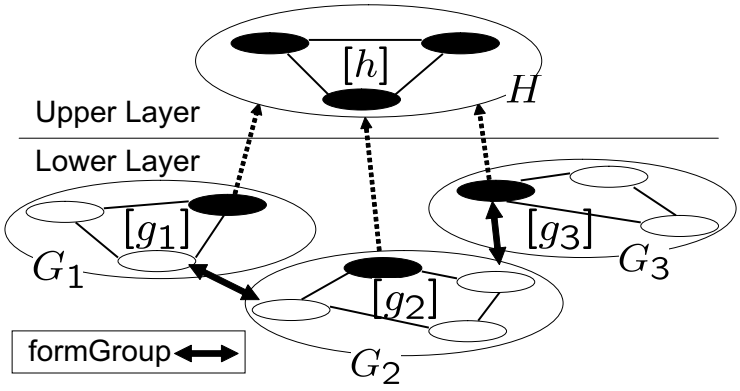


Figure 3.2: Formation of a hierarchical structure.

- *between network segments with large amount of traffic flow :* In MAN, some links between physically separate network segments may have large amount of traffic flow. Such links should be observed carefully in order for keeping stable traffic flow.

- *among network segments connecting to multiple providers/carriers :* A network segment may be attacked through several providers/carriers' links if it has multiple connected links.

The most of the above logical neighborhood relations are defined in advance. However, in some cases, the manager agents and/or sensor agents may want to specify the logical neighborhood relations dynamically when they are monitoring. In our technique, we have prepared a command for adding/deleting logical neighborhood relations dynamically.

3.2.3 Middleware for Describing Management Scenario

To describe management scenario easily, we provide several APIs for forming groups and group communication in our method. The syntax and semantics of the APIs are listed in Table 3.1.

formGroup(*channel***,** *cond***,** *maxsize***)** is the API to form a group among agents. **formGroup** does not block the behavior of the agent that calls it, that is, the procedure of group formation is executed in background of the foreground behavior specified after this API in the scenario. The group formation procedure

Table 3.1: Provided APIs.

API	Descriptions
formGroup(*channel*, *cond*, *maxsize*)	Requesting group formation with neighbor agents which satisfy condition *cond*. When the group is formed, a channel specified as *channel* is shared among group members. Multiple channels can be specified like "g, h". This API returns the number of member agents in the group.
setCommand(*channel*, *command*, *cond*)	Distributing a command specified by *command* for a monitoring item specified as *cond* to all members of the group via channel *channel*. used to change the monitoring items or conditions of packet filtering.
disconnect(G, S)	Member agents specified in set S leave from the group G.
$info$ = **getGrpInfo**(*channel*)	Obtaining the current management data from all member agents in the group via channel *channel*. A list of management data, $info$ is returned.
sendInfo(*channel*, *summary*)	Sending information *summary* via channel *channel* to one of group members. It is used to send the information to the manager agent or to the upper level group.
getInfo(*channel*)	Receiving information via channel *channel*. It is used with **sendInfo**.
numberOf(*cond*)	Obtaining the number of agents in the group which satisfy condition *cond*.

finishes when the agent forms a group or joins an existing group. The return value of **formGroup** is the number of agents in the group. When this API is executed first time, the value will be 1. Second or later time execution of this API returns the number of agents in the group to which the agent currently belongs. At the time, if the group formation procedure has already completed, a new group formation procedure is activated. **formGroup** is executed with condition *cond*, when an agent detects a problem identified by *cond*. If some of neighbor agents have executed **formGroup** with the same condition, a group is formed between those two agents. In a group, a channel specified by *channel* is shared among member agents. Member agents use the channel to exchange information. If **formGroup** is called without specifying a *channel*, a new channel will be created. Also multiple channels can be specified as a *channel* like "g, h". Each agent can execute **formGroup** repeatedly to accommodate other neighbor agents or other groups including neighbor agents. Consequently, the group can cover the area with the same problem. The number of agents in one group can be limited by the parameter $maxsize$. The agents periodically exchange the member information and they will not accommodate new agents or groups if the number of members exceeds the limit. If we need to make groups including more than $maxsize$ agents, the hierarchical connection among groups can be constructed. To make a hierarchical connection, **formGroup** is called with another *channel*. For example, assume that there are three groups G_1, G_2 and G_3 where the total number of agents exceeds $maxsize$ and the channels shared in those groups are g_1, g_2 and g_3, respectively, as shown in Fig. 3.2. If these groups execute **formGroup** by specifying channel h and the same condition *cond*, a new group H consisting of these groups are constructed. This hierarchy is useful to reduce control traffic for group communication in large groups.

In the parameter *cond* of the APIs **formGroup**, **numberOf** and **setCommand**, we can use primitives listed in Table 3.2. For example, conditions such as $warning.snort$ == MS_BLASTER (Snort detects MS Blaster) or $icmp.traffic > 3000000$ (icmp traffic exceeds $3Mbps$) can be specified. Other primitives not listed in Table 3.2 may be added by implementing them. Each primitive will return the value of specific data gathered from the segment watched by the agent. The condition *cond* can include any information that can

Table 3.2: Primitives used in conditions of APIs.

Primitives	Description
tcp.traffic(*pn*,*src*,*dst*)	Current traffic (in bps) of TCP flows. Information on traffic with specific port number, source IP address, and/or destination IP address can be obtained by specifying *pn*, *src* and *dst*, respectively. These parameters can be omitted.
udp.traffic(*pn*,*src*,*dst*)	Current traffic (in bps) of UDP flows. The parameters are the same as $tcp.traffic$.
icmp.traffic(*src*, *dst*)	Current traffic (in bps) of ICMP flows. The parameters are the same as $tcp.traffic$ except that *pn* cannot be specified.
ip.traffic(*src*,*dst*)	Current traffic (in bps) of IP flows. The parameters are the same as $icmp.traffic$.
throughput.traffic(*pn*, *src*, *dst*)	Current throughput (in bps). The parameters are the same as $tcp.traffic$.
anomaly.traffic	Anomaly detected in traffic. The return value is the type of anomaly. If the return value is 0, there is no anomaly.
warning.snort	Warning by Snort. The return value is the type of warning. If the return value is 0, there is no warnings.
id	The return value is the identifier of the agent.

be gathered from the segment by using SNMP or IDS.

setCommand(channel, command, cond) is the API to execute commands synchronously among all agents in the group. For the parameter *command*, one of the commands like $Watch$ for watching the traffic or $Filter$ for filtering specific packets can be specified. For example, when $Watch$ and $throughput.traffic$ (traffic in the segment) are specified as parameters *command* and *cond*, respectively, the monitoring item is changed to $throughput.traffic$, and after that all agents will collect information on this monitoring item.

$info$ = **getGrpInfo(***channel***)** is the API to collect the current management data from all agents in the group. The collected information is returned and

stored in an array $info$.

sendInfo and **getInfo** are the APIs to exchange information by 1 to 1 communication through *channel* between one of sensor agents in the group and the manager agent (or one of agents in another lower layer subgroup through hierarchy in Fig. 3.2). For example, assume that an agent executes $info$ = **getGrpInfo(***channel***)** to collect the management data from all the group members and then executes **sendInfo**(h, $info$). In this case, if the manager agent executes **getInfo**(h, $info$) the information included in $info$ (the collection of the management data from all the group members) will be sent to the manager. Instead of sending the whole information, the summarized information can be sent by using function *summary* to reduce the communication overhead. For example, we can easily implement function $summary(info)$ so that it composes a digest information such as the average, minimum, or maximum value for the current traffic. We have defined the semantics of **sendInfo** so that it finishes without sending information when the corresponding **getInfo** is not executed. So, by letting several group members execute **sendInfo**, they will try to send the information to the manager agent, and only one agent will succeed in sending, and the other agents will skip data transmission. This mechanism increases the system availability under network congestion and reduces control traffic to the manager agent.

disconnect(G**,** S**)** is the API which makes a subset S of member agents leave the group G. This API may be used after the problem solved.

3.2.4 Example of Management Scenario

With APIs and primitives in Table 3.1 and Table 3.2, we can easily describe scenarios for autonomous network monitoring. The management scenarios are written as script programs. We show an example scenario for sensor agents in Fig. 3.3. The behavior of this scenario is as follows.

1. Each agent which received MS Blaster warning from its local Snort tries to form a group with neighbor agents. If there are some agents which received the same warnings, a group is formed. Here, the maximum group member size is limited to 10.

2. If the number of members exceeds 4, the manager agent is joined to the

group, then each group member collects current traffic of each segment, summarizes it and sends to the manager agent. They execute a command by **setCommand** synchronously with other member agents and call **getGrpInfo** in order to collect the management data.

3. When member agents (i.e., the agents sharing channel g) which have observed traffic over 5 Mbps become the majority in the group, each member agent executes a command synchronously with other agents to change the current monitoring item to TCP packets with port number 135 (i.e., the port MS Blaster uses). The agent also executes a command to filter out the packets with port 135 when the proportion of those packets is more than 50% of the whole traffic. Then the agent calculates the proportion of these packets, summarizes the information and sends it to the manager agent.

3.3 Implementation of Middleware

3.3.1 Implementation of Autonomous Group Formation Function

We have implemented APIs in Table 3.1 based on our group communication middleware proposed in Refs. [89, 90].

When an agent A executes **formGroup** in its scenario, it sends message $GROUP$ including group formation condition and channel name(s) to all of neighbor agents which do not belong to the same group as A (see Fig. 3.4(a)). When neighbor agent B receives message $GROUP$ from A, it evaluates the group formation condition in the message. If the condition holds, B sends message ACK back to A. Otherwise, it does message $NACK$ (see Fig. 3.4(b)). A list of member IDs (denoted by $Members(B)$) of the group to which B belongs is attached to ACK. If A receives ACK, it composes message $CONF$ with member IDs (denoted by $Members(A)$) of the group to which A belongs, and sends the message back to B (see Fig. 3.4(c)). It also unifies $Members(A)$ and $Members(B)$, and keeps the unified member IDs as new $Members(A)$. When B receives message $CONF$ from A, it unifies two member lists and keeps the unified IDs as new $Members(B)$, as well. TCP is used for message exchanges between agents. By the above procedure, two groups (or agents) are combined

```
while(TRUE){
 warning_type = warning.snort;
 switch (warning_type){
 case MS_BLASTER:
   while(TRUE){
     num_of_members = formGroup("g,h",
        "warning.snort == MS_BLASTER", 10);
     if (num_of_members >= 5){
       setCommand("g", Watch,
        "throughput.traffic");
       info = getGrpInfo("g");
       formGroup("h", "id == Manager", 2);
       sendInfo("h", summary(info));
       if(numberOf("info >= 5Mbps")
            >= num_of_members/2){
         setCommand("g", Watch,
            "throughput.traffic(135)");
         setCommand("g", Filter,
            "throughput.traffic(135)
                >= 50%");
         info = getGrpInfo("g");
         sendInfo("h", summary(info));
       }
     }
   }
   break;
  case BAGLE:
   ...
  case MYDOOM:
   ...
  }
}
```

Figure 3.3: Example scenario for sensor agents.

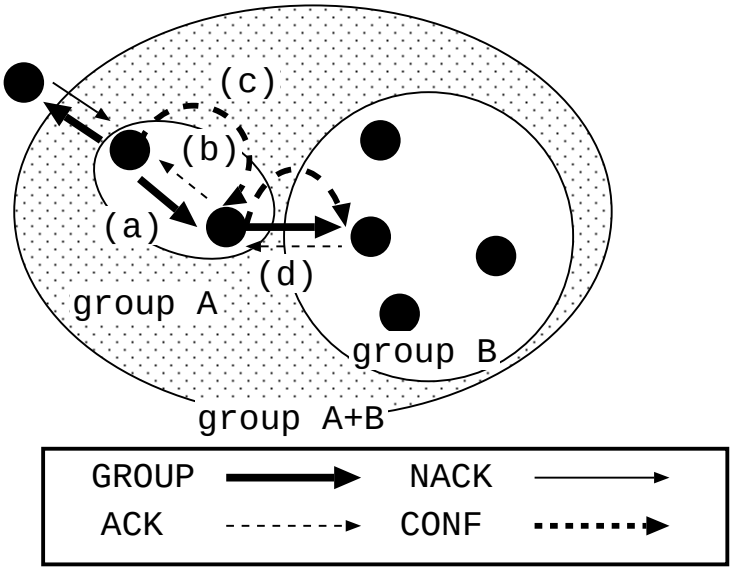


Figure 3.4: Formation of a group with our middleware.

into one bigger group (see Fig. 3.4(d)). Also, repeating the procedure several times, the size of a group is increased. We let each member of the group to keep the member IDs. When two groups are unified to one group, the member IDs are also unified if those two groups execute **formGroup** with the same channel name(s). The unified member IDs are shared among all members of the new group by multicasting the information with the technique used in implementation of **getGrpInfo**. At the same time, one *master node* is selected (e.g., based on minimum ID) among all members in the group. If the channel name(s) are different in two **formGroup**s which have been executed by two agents in two separate groups, these agents are selected as master nodes of those groups, and a member list consisting of these two master nodes is created and shared between them. Consequently, as shown in Fig. 3.5, the hierarchical structure called the *synchronization tree* is constructed among all members of a group. In the figure, circled nodes such as S_{11} represent the master nodes.

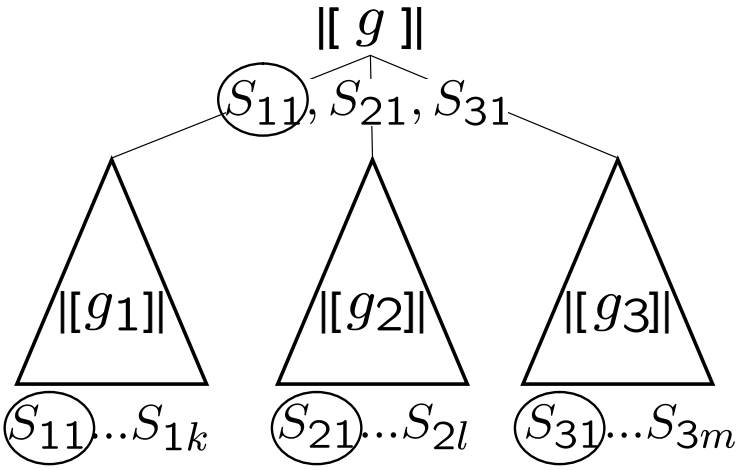


Figure 3.5: Synchronization tree.

3.3.2 Implementation of Group Communication Functions

As explained in Section 3.2, **setCommand** on channel g must be executed synchronously among all agents who share the channel g in the group. We implement this mechanism using the master node. When an agent A executes **setCommand**, it sends message $READY$ to the master node (denoted by M) to notify that A is ready to execute this API. The parameter values specified in **setCommand** are hashed into one value and this value is attached to the message so that only **setCommand**s with the same parameter values are executed synchronously among agents. When M receives $READY$ with the same hash value from all members in the group, it sends message $PERMISSION$ back to them. When each member agent receives this message, it executes the actual operation specified in **setCommand**. While network congestion occurs, $READY$ messages sent from some members may not reach M. In that case, M just ignores those members after certain time passed. If the master node M suffers from network congestion, we let neighbor node N to detect such a situation and to broadcast a message $REPLACE$ for notifying that N will be the new master node. When N receives ACK from all members, it serves as the new master node. If there are multiple neighbor nodes, we use the Bully algorithm [91] to decide the master node.

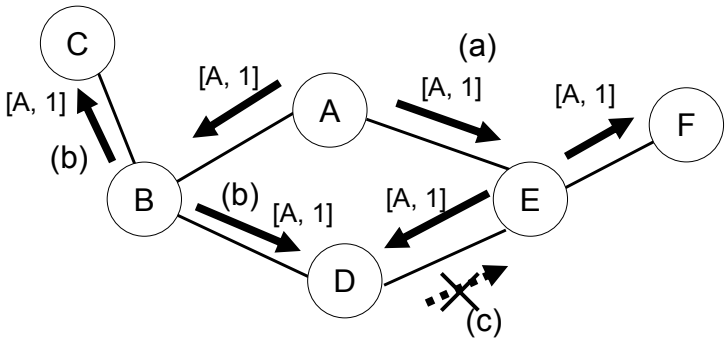


Figure 3.6: Flooding in a group.

getGrpInfo is implemented as follows. Each agent collects the monitoring information in its responsible network segment on the item specified by **setCommand**. When an agent A executes this API, it sends message $INFO$ with its management data to all of its neighbor agents in the group to which A belongs(see Fig. 3.6(a)). When another agent B receives $INFO$ from A, it forwards the message to all of its neighbor agents (except for A) within the group, as well. $INFO$ includes the ID of its owner agent and the sequence number used in the agent. Each agent which received $INFO$ checks the ID and the sequence number in the message to avoid forwarding the message more than once (see Fig. 3.6 (c)). Each agent which executed **getGrpInfo** is blocked until it receives $INFO$ from all members in the group or timeout occurs.

As we explained in Section 3.2, each pair of **sendInfo** and **getInfo** must be executed synchronously. We implement this synchronization mechanism with the master node as follows. First, when an agent A executes **getInfo**, it sends message $READY\_INFO$ to the master node M. We let M to have a FIFO queue called *demand queue* in it. When M receives $READY\_INFO$, it adds this message to the demand queue. After that, when another agent B executes **sendInfo**, it sends message $WRITE\_INFO$ with parameter values specified in this API to M. When M receives this message, it picks up the $READY\_INFO$ stored in the demand queue and forwards the $WRITE\_INFO$ to the sender of the $READY\_INFO$ as well as sends ACK to B. If there is no $READY\_INFO$ in the demand queue, M waits for a new message for a while and sends message $SKIPPED$ back to B if it receives no messages. The above mechanism achieves

both high system availability during network congestion and small control traffic around the administrator node, since multiple member agents (executing **sendInfo**) can try to send the same information to the administrator node, and only one $WRITE\_INFO$ message is sent to the administrator node.

3.4 Performance Evaluation

We have implemented APIs of the proposed middleware on the network simulator, ns-2, and evaluated the costs required for group formation and group communication such as the time used for group formation, total amount of control traffic, traffic used for group communication and so on. In order to show efficiency of the proposed technique, we have compared our middleware-based technique with the fixed hierarchical tree based technique such as Ref. [31]. The hierarchical tree based technique divides the target network into multiple segments, and places them along with a pre-defined hierarchical tree structure. The monitoring information is sent along with this tree. On the other hand, our technique autonomously forms a group with the same problem, and the monitoring information is transmitted by group communication among its group members.

3.4.1 Group Formation and Communication Costs

Configuration for Evaluating Group Formation and Communication Costs

In order to evaluate the basic performance, we have generated a hierarchical network topology using the topology generator *tiers* [92]. Table 3.3 shows the values of parameters. The delay time of each link was given at random from 1 msec to 20 msec, and the available bandwidth of each link was set to 100 Mbps. The transmission interval of $GROUP$ messages was changed among 0.1, 1.0 and 10.0 sec. The size of each control packet is 1,024 bytes if it does not have any member list. We used 1,024 bytes to represent a node in a member list with some information. For example, the size of the member list is 1,024×5 bytes when the number of members is 5. We set *true* for the parameter *cond* of **formGroup**, and activated agents to form groups at the same time. We varied the maximum size of members of a group from 1 to 50, and measured the time

Table 3.3: Simulation parameters.

CASE I : Basic Performance Evaluation	
# of nodes	50
Control packet size	1,024 ($bytes$)
Shared information size	1,024 ($bytes/node$)
Link delay	1.0–10.0 (ms)
Link bandwidth	100 ($Mbps$)
GROUP message interval	0.1, 1.0, 10.0 (s)
CASE II : Detection Performance Evaluation	
# of nodes	2,791
Control packet size	1,024 ($bytes$)
Shared information size	1,024 ($bytes/node$)
Link delay	10.0–20.0 (ms)
Link bandwidth	0.1–10.0 ($Gbps$)
GROUP message interval	0.1 (s)
Maximum size of group	10
Alert size of group	5
Attack traffic	1.6 ($Mbps$) (from one attacker node)
Threshold for group formation	5.0 ($Mbps$)

used for group formation, total amount of control traffic, traffic used for group communication and so on.

Costs for Group Formation and Group Communication

Fig. 3.7 shows the relation of the number of group members and the time necessary for group formation. From the figure, we see that the longer the interval of $GROUP$ messages, the time required for group formation becomes larger. However, the time necessary for group formation is proportional to $log\ N$ where N is the group size. As shown in the figure, the time necessary for group formation does not increase greatly as the number of group members increases if the transmission interval of GROUP messages is the same. On the other hand, it strongly depends on the transmission interval of $GROUP$ messages. For quick detection of problematic areas, the interval of $GROUP$ messages should be set between 0.1 and 1.0 sec.

Fig. 3.8 shows the relation between the maximum control traffic per link and the number of group members. From the figure, we see that the shorter the interval of sending $GROUP$ messages, the larger control traffic is generated for

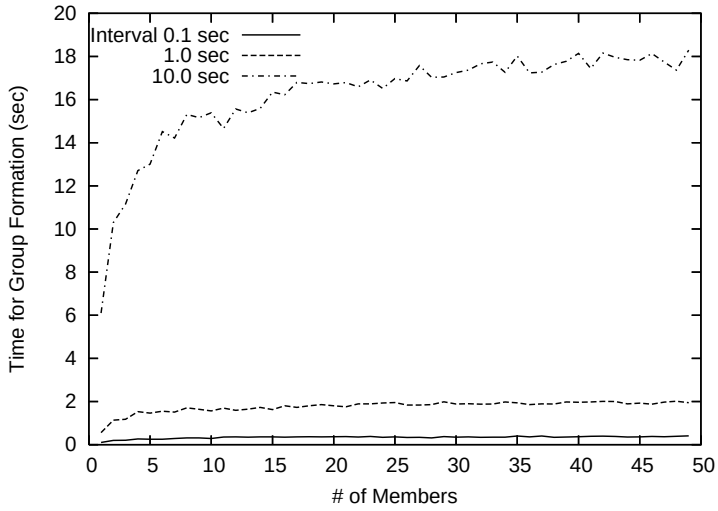


Figure 3.7: Time for group formation vs. number of group members (avg. of 20 simulations).

group formation. However, the traffic increases proportionally to the number of members, and is reasonably small (i.e., less than 100 kbps when the group size is limited to 10 members) even if we use a very short interval such as 0.1 sec.

Fig. 3.9 shows the traffic per link used for group communication where multicast data is transmitted from a node in the group. From this figure, we can see that the traffic per link is almost the same independently of the number of group members. Basically, the multicast data (e.g., INFO messages) is transmitted on each link at most twice. So, it does not depends on the number of group members.

From the above three figures, if the interval of sending $GROUP$ messages is set to less than a few seconds, the time necessary for group formation is small enough for detecting DDoS attacks. Also, we can see that if the maximum number of members of a group is limited to 20 or 30, we can keep the control traffic enough small.

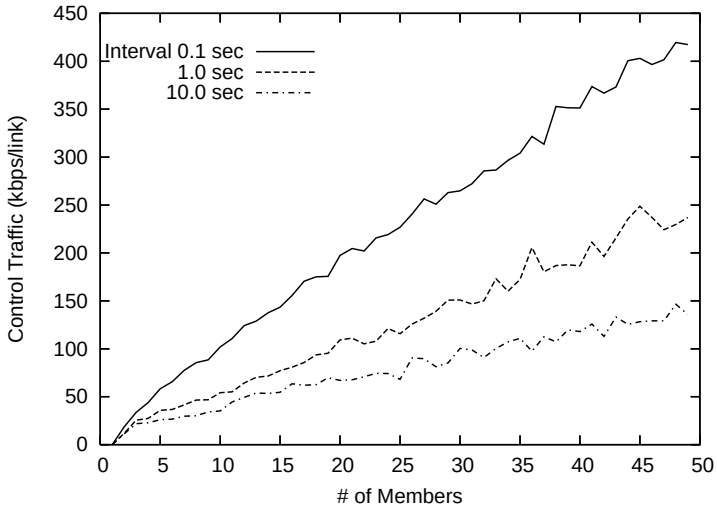


Figure 3.8: Number of members and control traffic (avg. of 20 simulations).

3.4.2 Detection Performance

Configuration for Evaluating Detection Performance

In order to evaluate the detection performance, we have used the topology of an academic network in Japan called the Science Information Network (SINET [93]) where there are 71 nodes and we have assumed that there are about 40 nodes in each site. That is, the number of all nodes is about 2,800. The values of parameters used for this simulation are also shown in Table 3.3. In this configuration, we have reproduced the situation that network congestion often occurs by a DDoS attack where about 500 nodes send UDP packets to a number of victim nodes (selected at random). Note that the number of the victim node was set to 1 in problem detection performance evaluation, and that it was set to 4 in DDoS detection performance evaluation. Attacker nodes were selected from leaf nodes at random and each attacker node generated 1.6 Mbps of traffic to the victim node. The time when each attacker node starts to attack was decided at random within 1.0–30.0 sec. To detect this DDoS problem, we have distributed sensor agents to all sites (71 sites) on SINET. The neighbor agents of each agent were specified as physically neighbor agents that can be reached by one hop on the experimental topology. In SINET, some sites have large

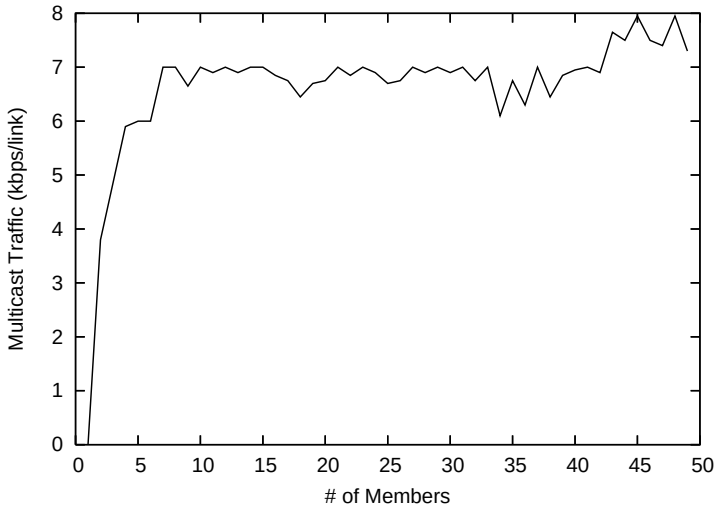


Figure 3.9: Traffic for group communication vs. number of group members (avg. of 20 simulations).

databases such as medical database, bio-informatic database, science database, engineering database and so on. Therefore, in our simulation, we have also given logical neighborhood relations among those sites.

We gave these sensor agents a management scenario. This scenario describes that each agent should form a group with neighbor agents when traffic to a certain destination exceeds the specified threshold (5.0 Mbps). Among the group members, the address list is exchanged. And they send the address list to the administrator node when the number of group members exceeds the alert size (5 was used). The interval for transmitting $GROUP$ messages was set to 0.1 sec. The maximum size of members of a group was set to 10. These values were decided based on the result of our preliminary experiments. With this scenario, we have reproduced the occurrence of problems caused by a DDoS attack, and measured the detection performance.

We have compared the result with the hierarchical tree based technique. In the hierarchical tree based technique, a static tree was decided based on the physical neighborhood relations among nodes, and the information of problematic areas is immediately sent toward the administrator node along the tree if each intermediate node of the tree has received the information from its child

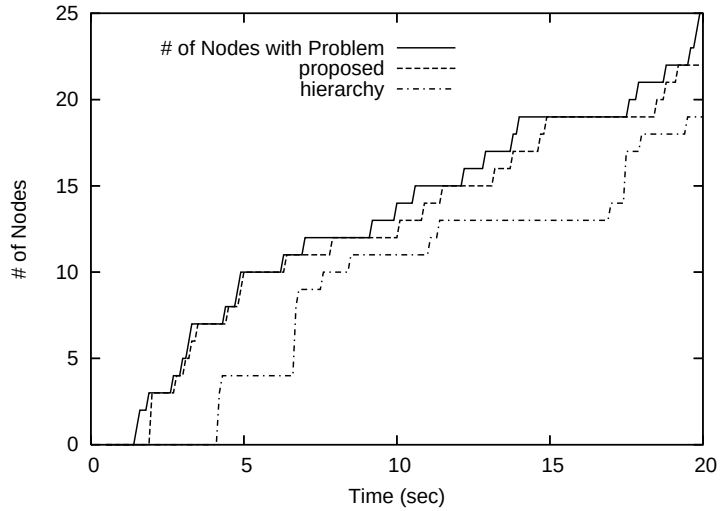


Figure 3.10: Time used for detecting problematic areas (without node failures).

nodes (i.e., sensor nodes) and the number of problematic nodes reported by those nodes exceeds the alert size (5 was used similarly to the case with our technique). Here, we let each intermediate node check every 0.1 sec whether it is attacked or not. When DDoS attacks occur, node failure may also occur. Therefore, in our simulation, we have reproduced the situations that some nodes failed and could not respond to any messages.

Problem Detection Performance

Fig. 3.10 shows the time used for the administrator node to detect problematic areas using the information from sensor nodes. Also, Fig. 3.11 shows the time used for the administrator node to detect problematic areas where we assumed that four node failures occurred. The four failure nodes were selected at random.

In the case without node failures, both our technique and the hierarchical tree based technique could detect the problematic areas. However, the detection time of our technique was smaller than the hierarchical tree based technique. As shown in the figure, the hierarchical tree based technique needs much more time than the proposed technique. For example, at time 5 sec, the problem is occurring at 10 nodes. Our proposed technique could detect all of these problematic nodes almost in real-time. On the other hand, the hierarchical

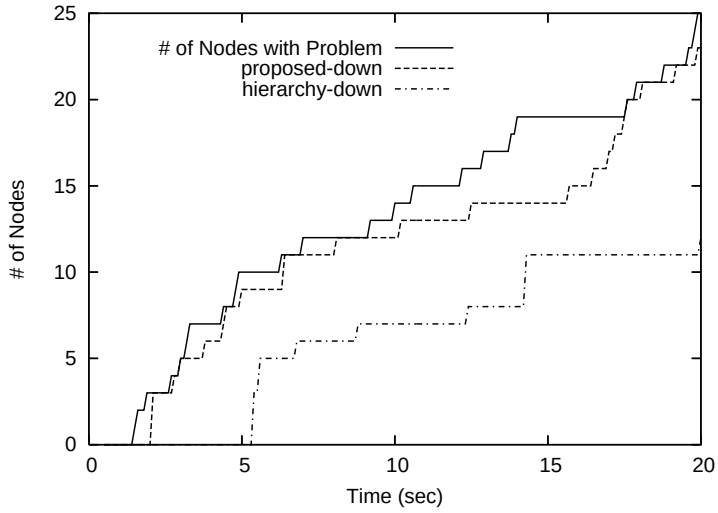


Figure 3.11: Time used for detecting problematic areas (with node failures).

technique detected them at time around 7.5 sec. This is because the hierarchical tree based technique uses only part of physical neighborhood relations among nodes as a tree, but our technique makes nodes to flexibly form a group by using all of neighborhood relations among them.

In the case with node failures, the hierarchical tree based technique could not detect all of the problematic nodes since some of intermediate nodes failed and they could not send the information to the administrator node. On the other hand, our technique could detect all the problematic nodes since all members of the group share the same information and any member can be the master node to send the information to the administrator node. Also, in our technique, the detection time was as fast as the case without node failures. Of course, if we use some backup nodes in the hierarchical tree based technique, and/or if we use multiple hierarchical trees for information delivery to the administrator node, the hierarchical tree based technique would be able to detect all of the problematic nodes. However, we believe that our technique is more useful in the sense that it does not need such an extra mechanism to detect problematic nodes efficiently.

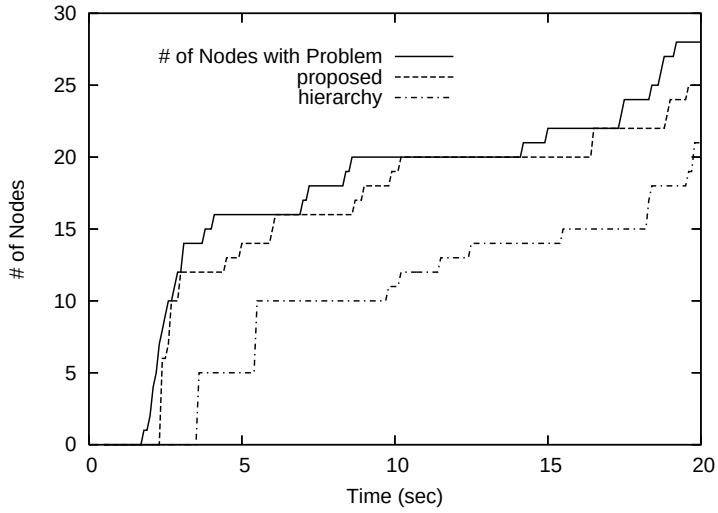


Figure 3.12: Detection of UDP flooding.

DDoS Detection Performance

Fig. 3.12 shows the time used for detecting UDP Flooding. In SINET, some sites have large databases, and we have given logical neighborhood relations among those sites. Here, we assumed that some of those physically separate network segments were attacked simultaneously.

In our technique, the administrator node could detect almost 100% of problematic areas immediately. This immediate detection was possible even for the early stage of attacks. On the other hand, the hierarchical tree based technique took a few seconds more time for detecting problematic areas than our technique. Especially, at the early stage of attacks, it is difficult for the hierarchical tree based technique to detect problematic areas precisely. In the hierarchical tree based technique, if physically separate network segments are attacked in parallel, and if the number of attacked network segments in physically neighborhood network segments is not so many, each intermediate node of the tree may not inform the problem to the administrator node so quickly. It may be only reported at the time when regular polling messages are sent. Our technique autonomously forms a group with the same problem using both physical and logical neighborhood relations. This reduces the time to detect problematic areas and makes quick reactive actions possible.

3.5 Conclusion

In this chapter, we proposed a technique for implementing distributed network monitoring systems based on autonomous group formation. In our technique, sensor agents are located over multiple network segments and they monitor and/or manage their responsible segments autonomously according to given management scenarios.

Through experiments, we showed that network monitoring based on our technique can detect the area where a problem is occurring in a few seconds with reasonable amount of control traffic and that our technique achieves higher availability of the management data for the administrator than the network management technique based on a fixed hierarchy model.

Chapter 4

Analysis of Mobility Effect on the Performance of MANET Applications

4.1 Introduction

In this chapter, we analyze the effect of mobility patterns on MANET applications and show a case study where link duration between neighbors are considered in DSR for stable ad-hoc communication. For the first issue, we think that the following services are examples of such new emerging applications, and they are affected by mobility.

- Communication services. Nowadays, wireless LAN devices are getting cheaper and smaller so that they can be embedded into cellular phones. A plausible story is that communication between two end nodes in the same local region is performed through mobile ad-hoc networks composed of the cellular phones equipped with wireless LAN devices and some fixed short range wireless stations. This will be effective to avoid waste of wide area network resources. Thus affect of mobility to the performance of mobile ad-hoc routing protocols directly influences the quality of the communication. In addition, several routing protocols based on mobility prediction have been proposed [94, 95]. The effects of mobility models, especially to such routing protocols, are not negligible.

- Information delivery services. Mobile ad-hoc networks may be used to

gather/distribute information such as advertisements in city sections, safety information for disaster relief and traffic congestion information advertisement to vehicles. In these services, in order to save the number of dissemination messages, position-based information dissemination may be used to deliver messages. In position-based information dissemination, each mobile node decides when it distributes the received message to the neighbors, depending on the (history of) positions of the node like geocasting [62, 63]. In this case, the node mobility may affect the message delivery ratio and the number of dissemination messages. For example, if we adopt a policy where only the nodes going to a shop distribute the advertisement of the shop, the people nearby the shop will receive the advertisement in a high probability, with the smaller number of messages than a simple dissemination where all the nodes distribute the advertisement to the neighbors.

- Mobile server services. Let us imagine mobile nodes have enough processing power and storage space to run server and database programs directly on the nodes. Then they may want to be servers (mobile servers) that provide the information stored on the nodes to the neighbors through ad-hoc networks. We call it *mobile server service*. A plausible situation is that a mobile node runs a weblog server program on the node, and the other nodes nearby the server node will access to the weblog through the mobile ad-hoc network. To realize this kind of services on mobile ad-hoc networks, content caching is a very important technology that allows the frequently accessed contents on a mobile server to be cached on the other mobile servers. This is effective to avoid traffic concentration and to improve content discovery ratio, because the mobile servers do not stay at the same positions. If contents of a mobile server are accessed frequently, but if the mobile server is going away from the current mobile ad-hoc network, the contents should be cached on other servers in neighboring locations.

To these services and applications, which are expected to be in widespread use, an evaluation technique with real environment should be provided. For this purpose, we have developed a wireless network simulator MobiREAL, which assists developers to model and reproduce the realistic deployment and movement

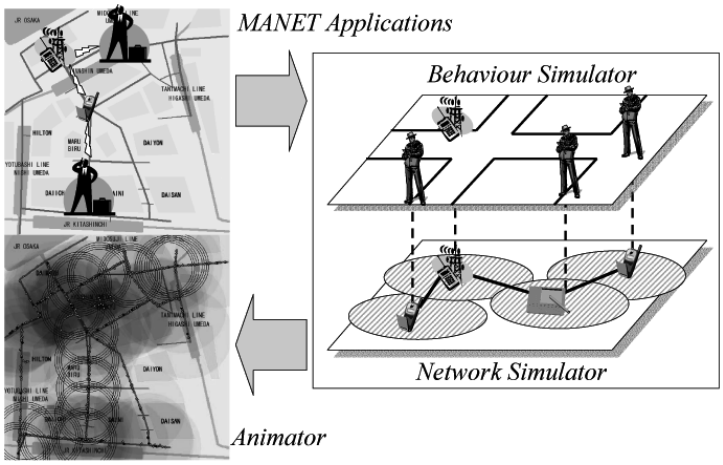


Figure 4.1: Architecture of MobiREAL simulator.

of network nodes.

The remainder of this chapter is organized as follows. Section 4.2 introduces the MobiREAL simulator. Section 4.3 presents quantitative analysis of random-based and realistic mobility models. Section 4.4 describes performance evaluation of several services/applications with modeling of real environment. Section 4.5 shows a case study in which link duration between neighbors are considered in DSR for stable ad-hoc communication. Finally, Section 4.6 concludes this chapter.

4.2 MobiREAL Simulator

MobiREAL has been developed in order to model realistic environment and to evaluate the performance of service and protocols in the environment. Fig. 4.1 shows the architecture of MobiREAL simulator. MobiREAL simulator is composed of two parts called *MobiREAL behavior simulator* and *MobiREAL network simulator*. We have extended the network simulator GTNetS [96] developed in Georgia Institute of Technology where dynamic node generation and deletion are newly supported and node positions, speeds and directions can be updated from the behavior simulator. For simulation of network and MAC layer protocols, our MobiREAL network simulator relies on GTNetS where DSR,

AODV and NVR (wireless form of Nix-Vector routing) are supported in the network layer and the standard IEEE802.11 DCF (CSMA/CA) with RTS/CTS is supported in the MAC layer. We have enhanced the signal propagation model of the original GTNetS so that obstacles are taken into account. It has a similar propagation model to the study of Ref. [47] where ground reflection is considered as zero propagation and line-of-sight is considered.

For our MobiREAL behavior simulator, a simulator user gives (i) a simulation field model, (ii) a set of behavior models of mobile nodes and (iii) a simulation scenario for generating mobile node objects. These are translated into the corresponding C++ classes using pre-defined library functions. For MobiREAL network simulator, the simulator user also gives C++ codes of network applications and user-specific protocols used in the simulation.

As (ii), users can specify the behavior of nodes which are dynamically changed according to the surroundings and the output from the applications. CPE model [26] is introduced for this purpose, and it can deal with interactive behavior with the network applications. In this model, nodes can change their paths and speeds dynamically according to the output from a network application, and the surroundings and status of the nodes. CPE model can simulate the interactive behavior with MANET applications, so we can reproduce various situations more realistically.

As (iii), users can use Urban Pedestrian Flows (UPF) mobility model [25]. CPE model describes the behavior of each mobile node, and this UPF model describes the behavior of pedestrians walking along streets. In Urban Pedestrian Flows, users specify the structure of the roads in a simulation field as a graph and the behavior pattern of pedestrians as a path on the graph. Also we assume that average densities of pedestrians on a part of streets in the field are given. These densities can be simply obtained by fixed point observations and so on. Then using a linear programming (LP) technique, for each path we calculate the number of pedestrians that follow the path per unit of time (called a *flow rate*), minimizing the maximum error between the observed density and that derived from the LP solution. Using the derived flow rates, we generate a UPF scenario which can be used in network simulators. An example of a mobility model derivation by UPF is shown in Fig. 4.2. The inputs are the potential paths (p_1, ..., p_5) and the observed densities of the edges. UPF calculates

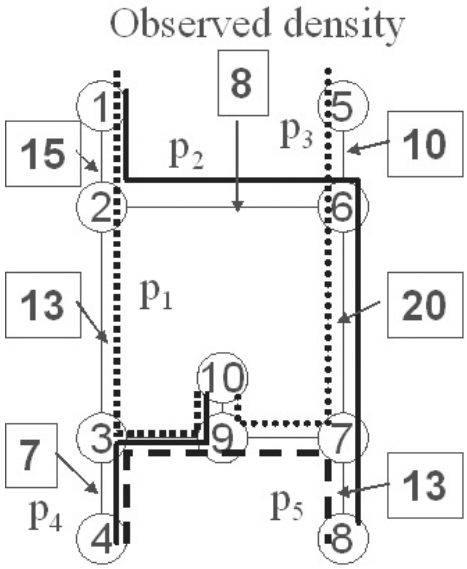
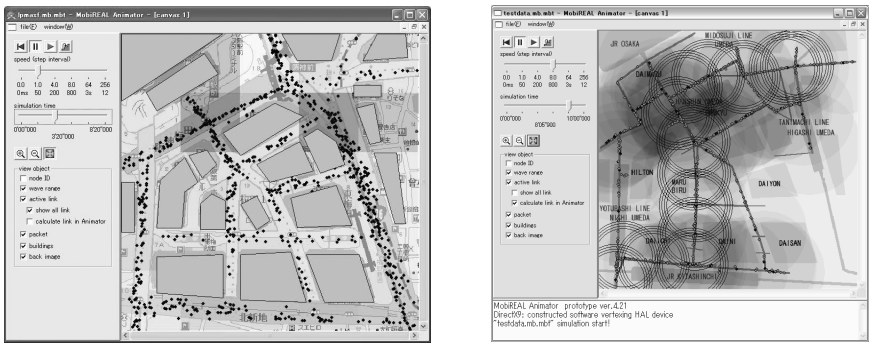


Figure 4.2: An example of UPF.

the flow rates of the given paths so as to make the flow rates nearly equal to the observed densities. For example, based on the observed densities, the flow rates of p_1, p_2 and p_3 are 13, 8 and 10 respectively. Then, there are some errors between the observed density on the edge 1-2 and 6-7. UPF minimizes the maximum of these errors.

The behavior and network simulators are two independent programs that periodically exchange necessary data through a TCP channel. When they are initialized, they hold the simulation field and node objects independently. The behavior simulator calculates the latest positions, directions and speeds of nodes, and sends them to the network simulator. Message sending events (i.e. inputs to networks) are also sent to the network simulator. The node objects in the network simulator hold their positions, directions and speeds. The network simulator conducts network simulation and packet exchange among those node objects is simulated based on the mobile nodes' signal propagation model.

MobiREAL UPF Scenario Generator In MobiREAL, users can specify a graph of the road structure through GUI to *MobiREAL UPF Scenario Generator*. If required information like pedestrian density is given, this tool generates



(a) UPF mobility model. (b) Transceiver application.

Figure 4.3: Animator snapshot.

a program code of a UPF mobility scenario that can be run in MobiREAL.

MobiREAL Animator MobiREAL Animator visualizes simulation traces of MobiREAL Simulator (see Fig. 4.3). Visualization of simulation results is one of the most important facilities for network simulators. In particular, visualizing mobility of nodes is mandatory for simulator users to confirm that the realistic behavior of mobile nodes is reproduced. Our MobiREAL animator runs on the Microsoft Windows platforms, and has been developed using DirectX 9. Each mobile node is represented as a small circle, and a semi-transparent concentric circle represents its radio range. Packet transmission is animated by concentrically growing circles with colors. To see an example demonstration of MobiREAL Animator, readers may refer to our MobiREAL web page [27].

4.3 Quantitative Analysis of Mobility Characteristics Using MobiREAL

We characterize a mobility model which can reproduce the walking paths of mobile nodes in real environments to learn the difference from the generic, random-based mobility models. The metrics used for comparison have been presented in Ref. [14], which characterize graph connectivity and mobility. These metrics are link changes (i.e. the number of link creations between two nodes), link duration (i.e. the longest time interval during which two nodes are in the transmission range of each other), degree of spatial dependence (i.e. similarity of velocities

of two nodes), degree of temporal dependence (i.e. similarity of velocities of two time slots of a node), and relative speed of two nodes. We have compared the Urban Pedestrian Flow (UPF) model with the Random Way Point with obstacles (RWP/ob) model.

4.3.1 Urban Pedestrian Flow Model

The UPF model is used to reproduce the pedestrian flows in city sections. Given (i) a simulation field represented as a graph G, (ii) a set P of paths on G where each path is an acyclic sequence of nodes of G, and (iii) observed densities of pedestrians on part of edges of G, this model can determine the *flow rate* of each path in P, minimizing the deviations between the derived and observed densities on the edges. Here the path rate is defined as the generation rate of pedestrians per unit of time at the starting point of the flow. In summary, this model can reproduce the realistic pedestrian flows from the observation results of densities of persons on some streets. Readers who are interested in the UPF model may refer to Ref. [25].

4.3.2 Random Way Point Model with Obstacles

We have defined a variant of the RWP model called RWP with obstacles model (denoted as RWP/ob). Given a simulation field graph G like the UPF model, each node randomly decides the next direction at each intersection, avoiding to go backward.

Comparison To generate a simulation field graph G, we have modeled a real 500m×500m region including buildings in the downtown of Osaka city. Then we have observed the pedestrians on a part of the streets for about a half of one hour to obtain the average density of pedestrians to generate the UPF mobility. The shortest paths were used as a set P of paths.

At first, we have measured (i) link changes (the number of link creations between two nodes) and (ii) link duration (the longest time interval during which two nodes are in the transmission range of each other). The distributions of these metrics are shown in Fig. 4.4 and Fig. 4.5. Clearly, the RWP/ob model has several cases with larger link changes (e.g. 9 and the above) compared with the UPF model. This is natural because in the UPF model, more neighboring

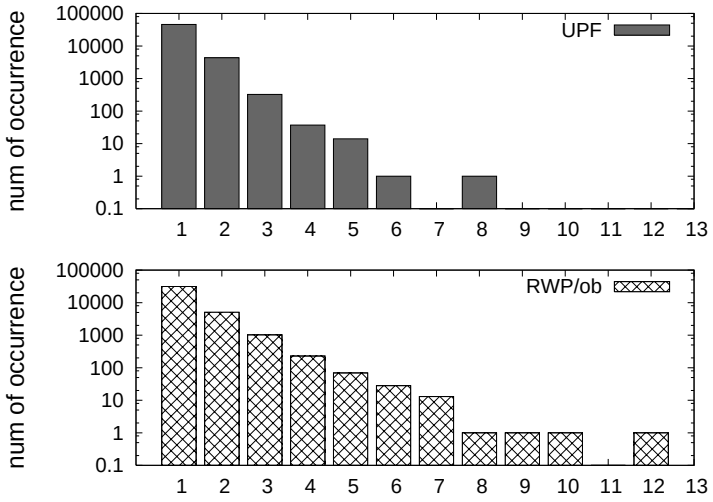


Figure 4.4: Distribution of link changes.

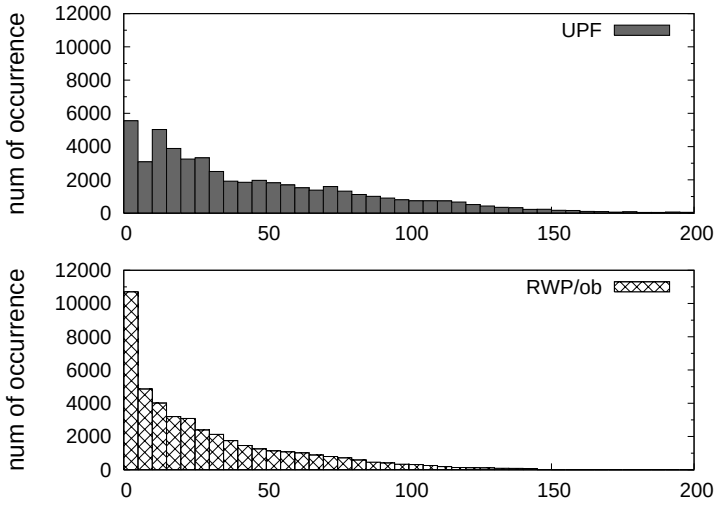


Figure 4.5: Distribution of link duration.

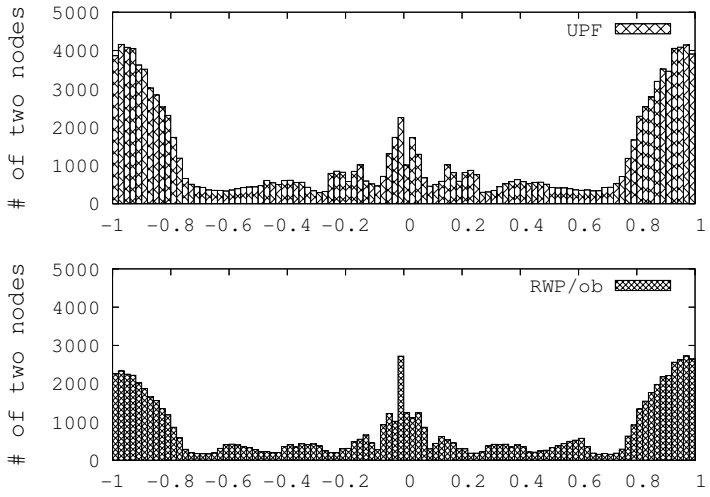


Figure 4.6: Degree of spatial dependence.

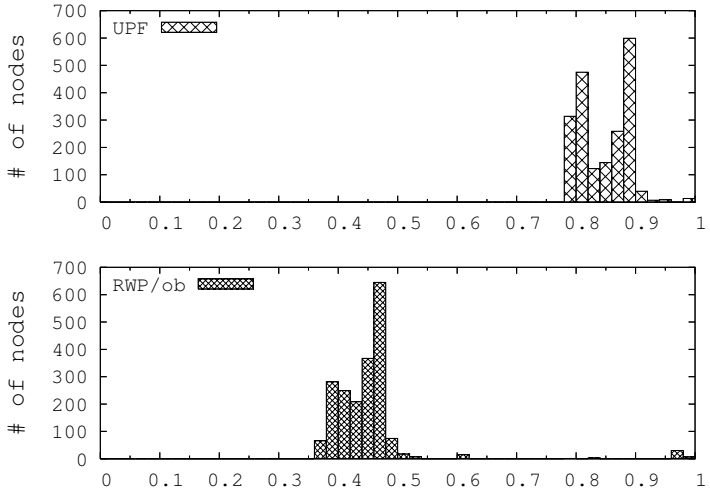


Figure 4.7: Degree of temporal dependence.

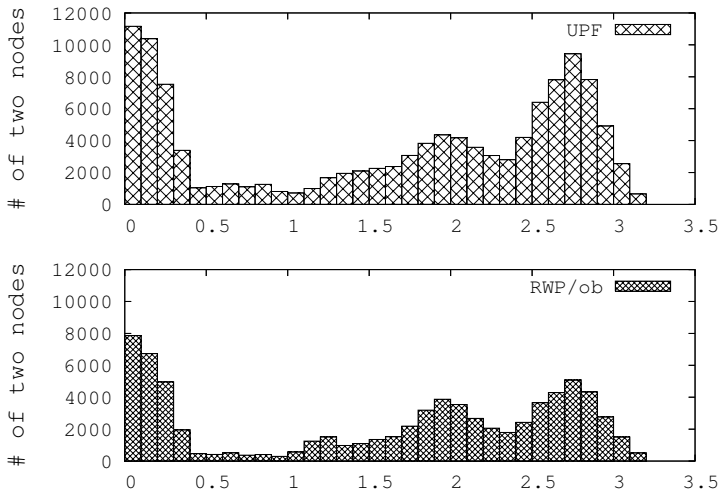


Figure 4.8: Relative speed of two nodes.

nodes are going to the same destination than the RWP/ob model and thus link changes do not occur many times. This observation is endorsed by the link duration result, where the UPF model has longer durations.

Secondly, we have measured the mobility characteristics metrics; (i) degree of spatial dependence (difference of velocities between two nodes), (ii) degree of temporal dependence (difference of velocities between two time slots of a node) and (iii) relative speed of two nodes. The results are shown in Fig. 4.6–4.8. In Fig. 4.6 and Fig. 4.8, the distribution patterns of the UPF model and the RWP/ob model are very similar with each other. Note that the total amounts of samples in the UPF model and in the RWP/ob model are different because there is a deviation on the density of nodes depending on places in the UPF model while it is almost uniform in the RWP/ob model. Fig. 4.7 shows the difference between two models. In the UPF model, each node does not change his/her direction at intersections frequently because each node goes to his/her destination along the shortest path. Consequently, the dependence in the UPF model becomes very high while it is not in the RWP/ob model.

4.4 Performance Evaluation with Modeling of Real Environment Using MobiREAL

We have simulated the services and applications as described in Section 4.1 with modeling of real environment by using MobiREAL. In this section, we analyze the results based on the analysis in Section 4.3. Design of related services exploiting mobility are also provided.

4.4.1 End-to-End Communication Service

We have measured the performance of DSR protocol on the UPF model and the RWP/ob model in order to see the influence of mobility on the quality of a communication service.

Simulation Settings

We have used the same simulation field graph G as the one in Section 4.3. The simulation time was 720 seconds. The communication was performed on UDP from time 240 sec to time 720 sec. The initial speeds of pedestrians are 2 m/s. After the connection was established, 10 kbps traffic was generated interactively between the two service users to simulate interactive real-time communication. We have used IEEE 802.11 DCF with RTS/CTS as the MAC layer protocol. The radio range was set to 100 meters. The service users were selected in the same pedestrian flow in the UPF model. In the RWP/ob model, they were selected randomly provided that the Euclid distance between the two becomes the same as the UPF model.

Experimental Results

The results are shown in Fig. 4.9–4.12. In the UPF model, the route was established along the flow. Therefore, the route is more stable than that in the RWP/ob model. This observation is endorsed by Fig. 4.9 and Fig. 4.10. The packet arrival ratio becomes lower and the number of control packet is larger if the route is instable, since there are many disconnections of the route. In Fig. 4.11, the UPF and RWP/ob models have similar patterns because the distances between two service users were almost the same in both models. Finally in Fig. 4.12, the RWP/ob model has similar hops in many cases because the density on

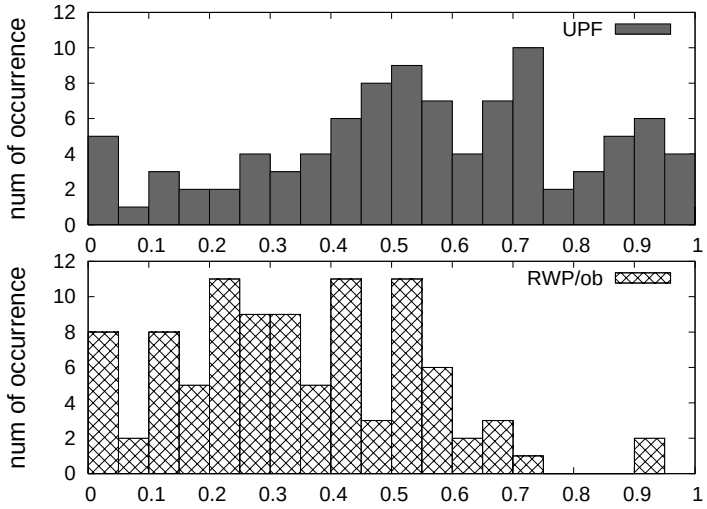


Figure 4.9: Distribution of packet arrival ratio (captured during every 5s and the averages per second are shown).

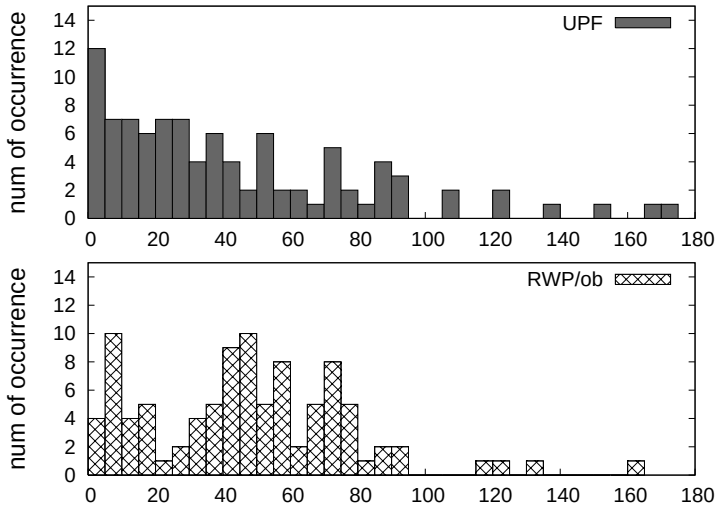


Figure 4.10: Distribution of number of DSR control packets (captured during every 5s).

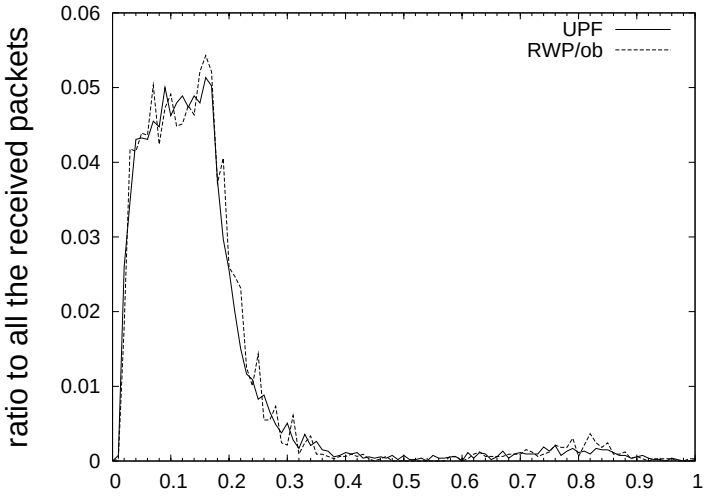


Figure 4.11: Distribution of packet delay.

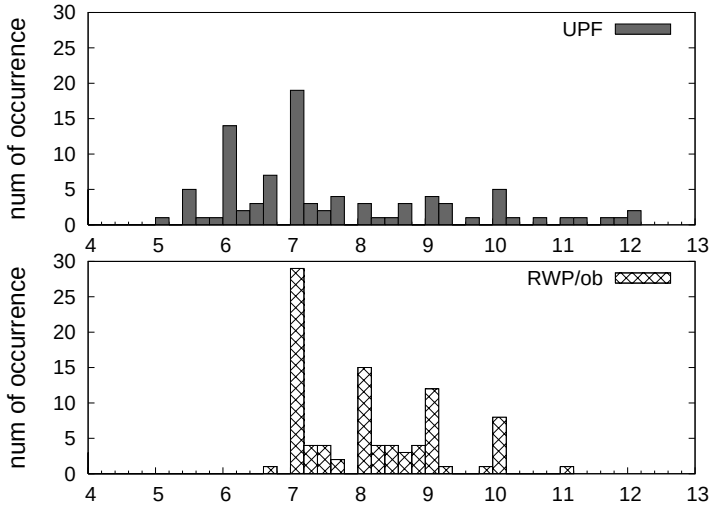


Figure 4.12: Distribution of DSR hops (captured during every 5s and the averages per second are shown).

any street was almost identical. On the other hand, the hops were different in the UPF model because each street had a different density.

From the results, it is obvious that such a link selection policy may be suitable that selects stable links. The "stability" can be estimated based on mobility characteristics which we presented in Section 4.3 or probabilities as done in Ref. [97]. Also the links with long durations will be selected in route discovery. In Section 4.5, we present a technique to estimate stability of links based on mobility prediction.

4.4.2 Information Dissemination Service to Pedestrians

We have measured the performance of an information dissemination service. This dissemination is performed through a mobile ad-hoc network where each mobile node caches a message received from a fixed station or a neighboring node. After receiving the message, it checks for every 15 seconds whether it should send the cached information to its neighbors or not, according to the predefined probability function $Prob(P, S)$. This is called *dissemination policy*, where P and S denote the positions of the node and the base station, respectively. After 60 seconds or once it sends the cached information, the node discards the information. Therefore, each node has at most 4 chances to send information. As the dissemination policies, we provide the following three $Prob_i$ $(i = 1, 2, 3)$;

1. $Prob_1(P, S) = \frac{d_{max} - d(P,S)}{d_{max}}$ where $d(P, S)$ is Euclid distance between P and S and d_{max} is the maximum Euclid distance of the field (in our simulation field $d_{max} = 500\sqrt{2}$). This dissemination policy indicates that if the node is located closer to the base station, it broadcasts the information with higher probability.

2. $Prob_2(P, S) = \frac{d(P,S)}{d_{max}}$. This is the opposite of the dissemination policy $Prob_1$.

3. $Prob_3(P, S) = 0.5 * C(d(P, S) \leq d_{const})$. The value of $C(t)$ is 1 only if the condition t is true, otherwise 0. We have set $d_{const} = 150$ m. This dissemination policy means that the node broadcasts the information with probability 0.5 if the distance $d(P, S)$ is no longer than 150m.

Simulation Settings

Most settings follow the previous example except that the simulation time was 600 seconds, and the dissemination was performed from time 180 sec. to 600 sec. The base station was placed near the center of the field.

Experimental Results

We have measured the information dissemination ratio, which indicates how many nodes obtained the information. Fig. 4.13 shows the result. Depending on the dissemination policies, the dissemination ratios are different, but the UPF model and the RWP/ob models have similar patterns. Here, we can see that the ratio in the UPF model is better than that of the RWP/ob model. This is because the UPF model has pedestrian flows, thus once a node obtains a message, the others on the same flow are easily "infected".

On implementing such a service in urban areas, we should design the dissemination policy being aware of pedestrian flows if possible. Since the information is easily distributed to the nodes on the same pedestrian flow, we should adopt a policy to let messages jump to the other flows to disseminate the messages quickly. If the history of node mobility is available on each node, such a policy may be effective that sends a message when a node discovers another node that is expected to go to a different direction.

4.4.3 Information Dissemination Service to Vehicles

As the third example, we have measured the performance of an inter-vehicle ad-hoc communication protocol called RMDP/CA, which disseminates the preceding traffic information to the following vehicles. This protocol is extended from RMDP presented in Ref. [98].

Similar with the second example, each vehicle disseminates traffic information received from a neighboring vehicle to its neighbors. Each vehicle holds a certain number of traffic information, and at relaying received information, the node merges the received information with its own information.

In order to increase message delivery ratio, the dissemination interval is determined as follows. Generally, in the low density regions, the dissemination interval should be short and in the high density regions (in case of traffic jams,

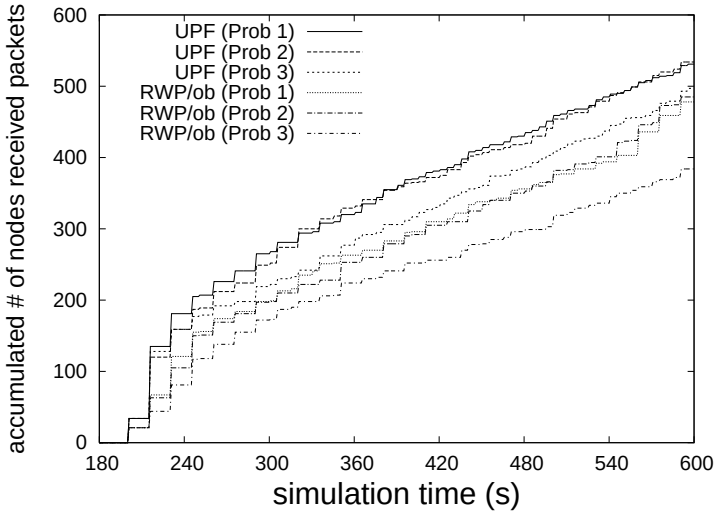


Figure 4.13: Performance of information dissemination service to pedestrians.

for example) the dissemination interval should be long to avoid message collision. Ref. [98] has proposed Received Message Dependent Protocol (RMDP) where users set the dissemination interval to the inverse proportion to the number of received messages. We have also decided concrete dissemination intervals based on simulation results so that we can maximize the message delivery ratio. We note that if collision occurs frequently, the numbers of messages received by vehicles successfully become small. This makes the dissemination interval of RMDP shorter because from the number of received messages, each vehicle cannot distinguish low density regions from high density regions with communication errors. To cope with this problem, we have designed RMDP/CA (RMDP with Collision Avoidance) where collision detection functionality is assumed as the MAC layer.

We have used two mobility models of vehicles, (a) random way point based mobility model and (b) traffic flow based mobility model. They are denoted as the RWP model and the TF model respectively, and explained below.

Simulation Settings

We have used NETSTREAM traffic flow simulator [99] developed by Toyota Central R&D Lab., Inc. in order for performing the effective prediction for

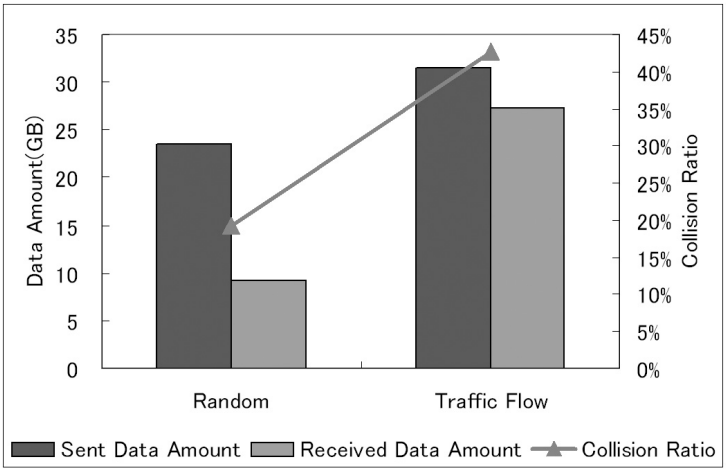


Figure 4.14: Performance of information dissemination service to vehicles.

traffic jams and prior evaluation of ITS introduction. It defines traffic flow characteristics and the lengths of signals for all road links based on real load maps and signal conditions, and then calculates all vehicles' movements at every second. It could rather exactly predict the traffic jams when Nagano Winter Olympic was held.

We use a real city map (20 km square) and its signal conditions, and estimate the number of vehicles on each road based on its survey. The number of vehicles was about 4,400. In the RWP model, we randomly specified the starting points and destination. Its traffic flow was basically flat. Each vehicle does not wait for the signals. It only follows the road when it decides its direction of movement. It is close to the Manhattan mobility model and the city section mobility model [100]. It cannot reproduce traffic jams although it can consider the density of vehicles. On the other hand, in the TF model, traffic amounts are different depending on roads, and traffic jams have occurred near signals with heavy traffic.

We assume that each vehicle has IEEE 802.11 based wireless communication facility. The size of the disseminated information was 1 Kbyte.

Experimental Results

Fig. 4.14 shows the ratios of transmitted data amount and received data amount and the collision ratios under the two mobility models. We note that in this figure, if two vehicles receive the same data sent from a vehicle, the number of the received data amount is counted as two. Since the node density was rather low in the RWP model, the received data amount and the packet collision ratio are rather low. On the other hand, in the TF model, there were some traffic jams and thus the density of some regions was high. As a result, the received data amount and packet collision ratio are rather high.

These results do not suggest the importance of mobility but we can see the impact of node deployment to the performance of the service.

4.4.4 Information Collection for Disaster Relief

Mobile ad-hoc networks will be useful in case of emergency such as disaster relief. As the last example, we have measured the performance of a service which can be used to collect "alive messages" (safety informations) from people in a disaster-stricken region.

This service assumes pedestrians (victims in this example) and vehicles as mobile nodes. Through the mobile ad-hoc network which is composed of those nodes, the safety information from each victim is flooded and arrives at a shelter eventually, at which the safety informations of the victims are accumulated. Inter-vehicle communication is controlled by RMDP/CA like the third example.

We have used two mobility models. The one is the RWP model. Because there are pedestrians and vehicles, we have merged the RWP/ob model used in Section 4.3 and the RWP model for vehicles used in the third example. Another is called the disaster model (denoted as the DS model) where vehicles move along the roads and follow the traffic rules while some victims walk around and some others walk toward the shelter.

Simulation Settings

We have modeled a 1km×1km region of a disaster-stricken area includes a base station (shelter) in the center. We have assumed 3,000 victims and 1,000 cars in the region equipped with short range wireless radio devices. The radio ranges were set to 100m for pedestrians and 200m for vehicles.

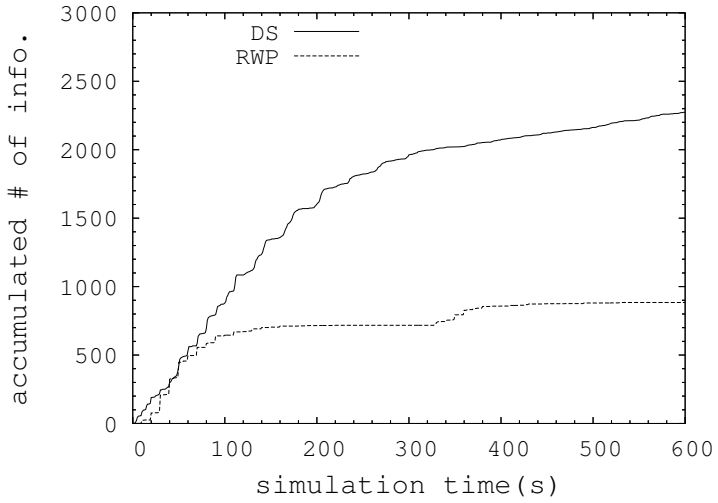


Figure 4.15: Performance of information collection system for disaster relief.

Experimental Results

Fig. 4.15 shows the number of safety informations of victims accumulated to the shelter under the two mobility models. In this result, the number of informations increases faster and continues to grow in the DS model while that in the RWP model reaches the peak earlier. The reason of this difference seems the same as the second and the third examples. In the RWP model, because the nodes are uniformly distributed on all over the region, the number of nodes nearby is relatively small. On the other hand, with the DS model, because pedestrians move in crowds toward the shelter and vehicles move along the roads, the information is disseminated along the flow easily.

4.5 Effect of Mobility Consideration on MANET Protocols

4.5.1 CDE: Connectivity-based Direction Estimation

The basic principles of CDE are to monitor connectivity between neighbors and estimate the freshness and duration of residence time in the wireless range. We consider that a communication link is more stable when its duration is longer

because the node goes into the same direction and its relative speed is low.

CDE algorithm is as follows. Each node periodically broadcasts a beacon to its neighbors at interval T. Hence, the time-line is divided into time slots of length T, and each node broadcasts one beacon per slot. Note that global time is not required since we need only local timers in each node. This beacon includes the sender's identification. Hereafter, $b_j(t)$ denotes a beacon from node j at slot t. Each node i maintains its *neighbor table* N_i, which includes *active ratios* of other nodes. The active ratio of node j means how long j exists around the node. Hereafter, we denote an active ratio of node j maintained by node i as $ar_i(j)$. In order to calculate active ratios, each node i maintains j's *active count* $c_i(j)$ $(0 \leq c_i(j) \leq W)$, which is the number of successfully received beacons from node j in the last W slots at node i. Then, node i can periodically calculate:

$$ar_i(j) = \frac{c_i(j)}{W}$$

Moreover, CDE calculates *short active ratios*, which are active ratios in the latest short term. A short active ratio of node j at i is denoted as $sar_i(j)$ and maintained in a similar way to $ar_i(j)$ by using the number of beacons $sc_i(j)$ received in the last W_s slots.[1] Thus, i also periodically calculates:

$$sar_i(j) = \frac{sc_i(j)}{W_s}$$

The followings are detailed procedures of nodes for calculating (short) active ratios and node density. Node i records the receipt of $b_j(t)$ in N_i when it receives a beacon $b_j(t)$. When the time slot t ends, node i checks each entry of node k in N_i and increases $c_i(k)$ if i has received $b_k(t)$. Otherwise, $c_i(k)$ is decreased. At the same time, i counts the number d_i of received beacons in the last slot t in order to estimate the node density around it.

Each node i maintains the latest $ar_i(j)$, $sar_i(j)$ and d_i as described above. By using these metrics, CDE enables to form stable ad-hoc links. Some design examples are given in the following section.

4.5.2 Modification of DSR Protocol Using CDE

DSR

Dynamic Source Routing (DSR) [60] is one of well-known routing protocols on MANETs. We describe the basic process of DSR here. When node s does not

[1] We assume $W_s < W$ and obviously, $0 \leq sc_i(j) \leq W_s$.

know a route to node d, multihop routes are established as follows:

1. s floods a Route Request message (RREQ) to the network (RREQ phase).
2. d unicasts a Route Reply message (RREP) to s when it receives RREQ (RREP phase).

In the RREQ phase, each node adds its own identification at the tail of the node sequence contained in the received RREQ and broadcasts RREQ when it receives RREQ. Then, when the destination d receives RREQ, d checks the route from s (e.g., the sequence of identifications). d simply reverses the route and returns RREP to s through the reversed route in the RREP phase if bi-directional links are assumed.[2] Each node caches the routes to s when it receives RREP. A route cache of each node is also updated when it receives RREQ or overhears neighbors' transmission.

In case of route breaks, a Route Error message (RERR) is sent to s. For example, assume a route from s to d as $\{s, t, u, d\}$. u sends RERR to s via t when the communication link $u - d$ has been broken. s floods RREQ again to find another route or uses another route if available in its route cache. Also, if t knows another route to d when t receives RERR from u, t re-routes by using its cache and notifies the route change to s. This route recovery function using caches is useful since it reduces message overhead for RREQ flooding.

DSR using CDE

We give a case study where CDE is applied to DSR. For simplicity of discussion, bi-directional communication links are assumed here. However we can easily apply the following discussion to the uni-directional case.

Again, our goal is to establish stable communication links. For this purpose, we modify DSR so that each node discards RREQ received via unstable links. In DSR, RREP and RERR are sent through the routes where RREQ has been forwarded by flooding. Thus, restricting flooding routes of RREQ also restricts routes of RREP and RERR.

Here, we introduce three threshold values AR, SAR and D to represent active ratios, short active ratios and node densities, respectively. When these

[2]In realistic environment, radio range fluctuates and is not uniform. We may not be able to assume bi-directional links hence. In such a case, d needs to find routes to s similarly by flooding RREQ.

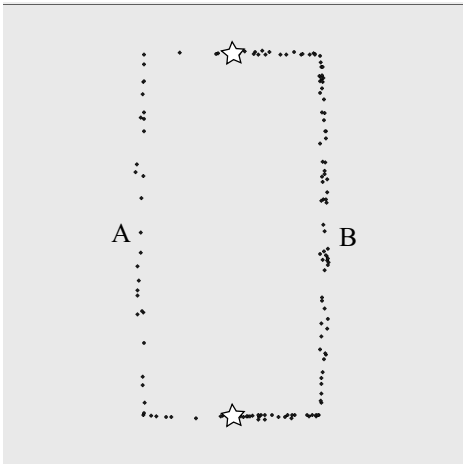


Figure 4.16: Simulation map.

Table 4.1: Simulation parameters.

Radio range	100 (m)
Bandwidth	1 $(Mbps)$
Moving speed	Normal dist. of $\mu = 1.5$ (m/s), $\sigma^2 = 0.1$
Beacon interval	1 (s)
Data size from users	5,120 $(bytes)$
# of nodes	154
Street length	500 (m)

metrics of node j are updated, each node i checks if the value of each metric ($ar_i(j)$, $sar_i(j)$ or d_i) is greater than its threshold and sets a *valid bit* to the entry of j in neighbor table N_i if all the metrics satisfy the condition. Otherwise, i sets an *invalid bit* to the entry of j. Based on the valid bit, when i receives RREQ from j, it decides whether it discards RREQ or not. That is, when i receives RREQ from j, *i processes the RREQ phase if and only if the valid bit of j is set.* In the next section, we show that this simple policy effectively supports to establish stable communication links in DSR.

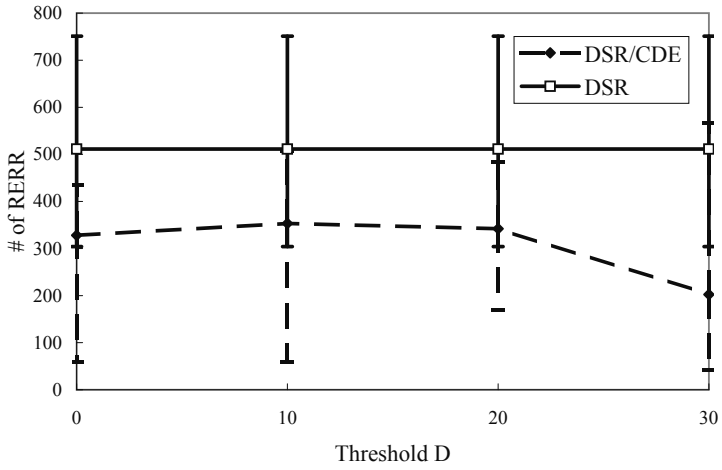


Figure 4.17: The number of RERR vs. threshold D.

4.5.3 Performance Evaluation

Simulation Settings

We have used a map which has two streets as shown in Fig. 4.16. Each street has two corners and their end points are connected via the points where stars are located in the map. We located two fixed nodes at the star points. These two nodes communicate interactively by UDP over the MANET using the DSR protocol. In this scenario, user i starts to send voice data of 5,120 bytes to another user j. Then, when j receives the data from i, j sends voice data of 5,120 bytes to i. This interaction is repeated. For detailed simulation settings, see Table 4.1. We have set threshold values of AR and SAR to 0.8 and 0.5, respectively and measured the performance of DSR changing the value of D. In the simulation, DSR using CDE (DSR/CDE) is compared with the original DSR.

In order to confirm the effect of different node densities, the average node density on street B has set to about three times that on street A. Each node continues to move between the two end-points of the initial street.

Results

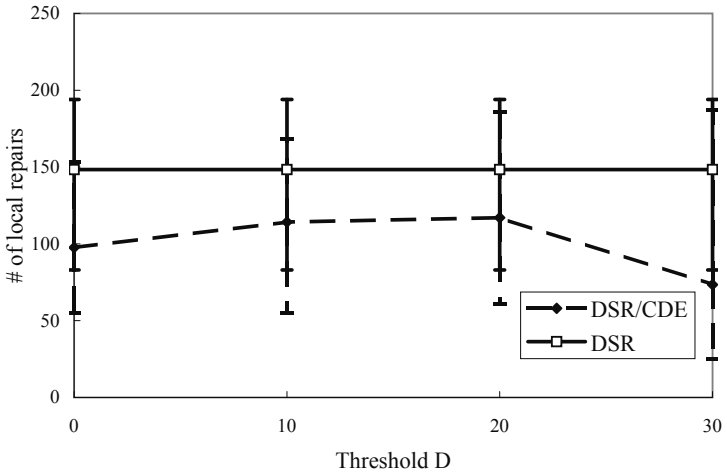


Figure 4.18: The number of local link repairs vs. threshold D.

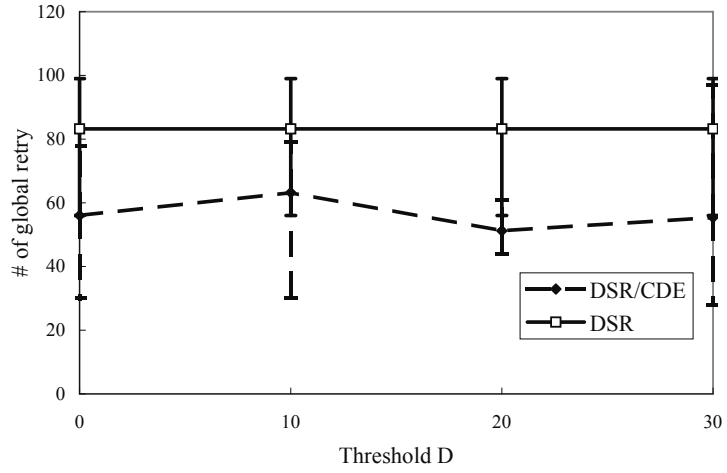


Figure 4.19: The number of global link repairs vs. threshold D.

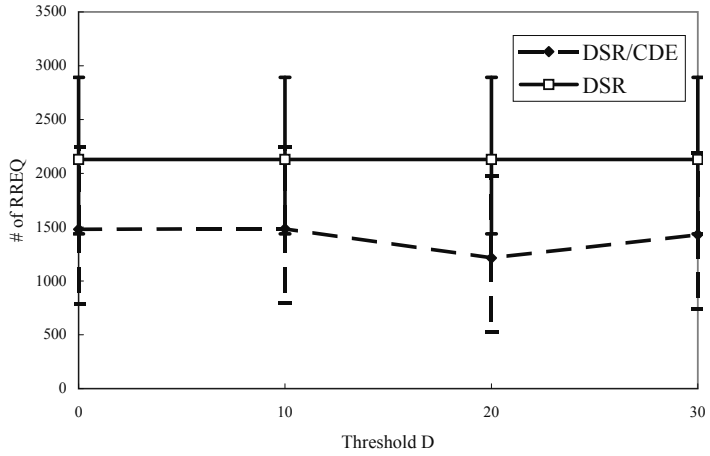


Figure 4.20: The number of RREQ vs. threshold D.

Stability of Routes Fig. 4.17 shows the number of RERRs. Obviously, DSR/CDE achieves better performance than DSR. This result means that link breaks in DSR/CDE occur less than DSR(other words, CDE enhances stability of DSR). Moreover,the number of local repairs and global repairs is shown in Fig. 4.18 and Fig. 4.19, respectively. These results also show that the number of link breaks in DSR/CDE is less than that in DSR and remark the effectiveness of CDE to increase stability of ad-hoc communication. Also in Fig. 4.20, we can see that the number of RREQs in DSR/CDE is less than that in DSR because CDE enhances the stability of links.

Communication Performance Fig. 4.21 shows the number of messages sent from the two fixed nodes. The number of sent messages in DSR/CDE is slightly less than that of DSR when $D = 30$. This is because D is sometimes too high and the number of candidate links becomes small. We can see that DSR/CDE achieves better performance in packet delivery ratios (see Fig. 4.22). In Fig. 4.21 and Fig. 4.22, we can see that DSR/CDE can establish more stable communication links since packet delivery ratios in DSR/CDE are higher than DSR when almost the same number of messages are sent. In Fig. 4.23, the average delay in DSR/CDE is very similar to that in DSR except in the case of $D = 30$ again. Therefore, we need to set thresholds carefully or may need some

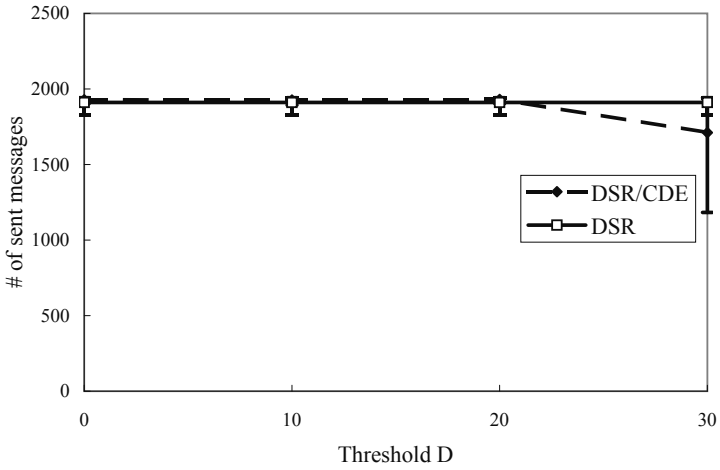


Figure 4.21: The number of sent messages vs. threshold D.

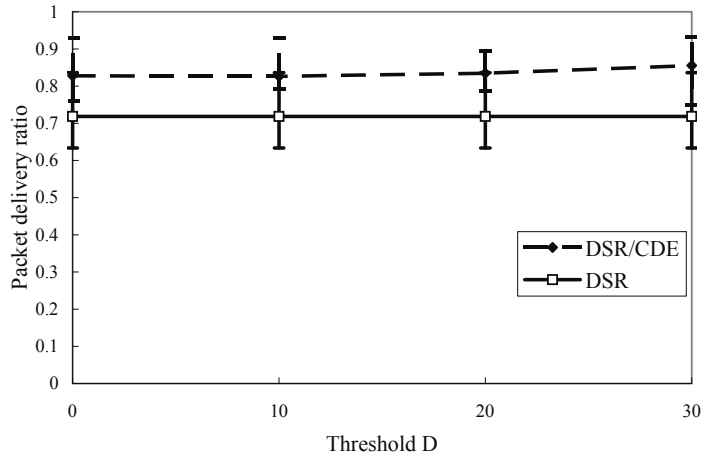


Figure 4.22: Packet delivery ratios vs. threshold D.

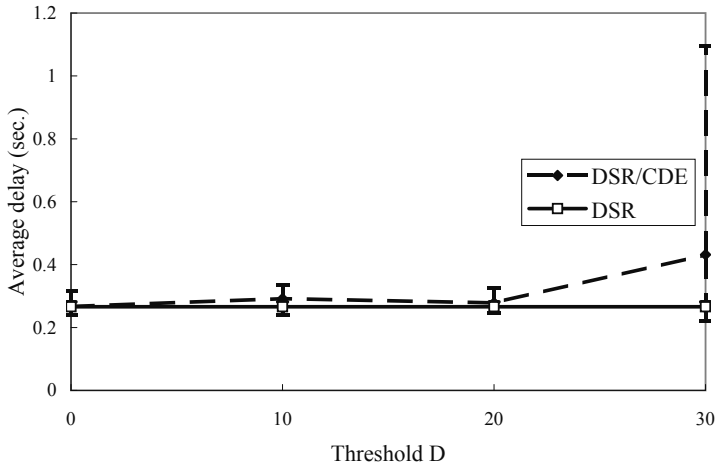


Figure 4.23: The average delay vs. threshold D.

autonomous threshold tuning mechanisms. However in most cases, DSR/CDE shows better performance and thus CDE improves stability of DSR.

4.6 Conclusion

In this chapter, we have analyzed the effect of mobility models on MANET applications through simulation using MobiREAL simulator and confirmed significance of considering realistic deployment and movement of pedestrians and vehicles in the design of mobile ad-hoc network services in urban areas so that these services can achieve high performance and quality in such environments. Based on this analysis, the Connectivity-based Direction Estimation (CDE) has been proposed and is applied to DSR. The experimental results have shown that stability of ad-hoc communication can be increased by considering characteristics of node movement.

Chapter 5

Ad-hoc Localization in Urban Districts

5.1 Introduction

In this chapter, we present a localization algorithm called UPL (Urban Pedestrians Localization) for positioning mobile nodes in urban districts. The design principle of UPL is two-fold. (1) We assume that location seeds are deployed sparsely due to deployment-cost constraints. Thus most mobile nodes cannot expect to meet these location seeds frequently. Therefore, each mobile node in UPL relies on location information received from its neighboring mobile nodes in order to estimate its area of presence. The area of presence of each mobile node becomes inexact as it moves, but it is helpful to reduce the areas of presence of the other mobile nodes. (2) To predict the area of presence of mobile nodes accurately under mobility, we employ information about obstacles such as walls, and present an algorithm to calculate the movable areas of mobile nodes considering obstacles. This also helps to reduce each node's area of presence. The experimental results have shown that by the above two ideas UPL could achieve 8m positioning error in average with 10m of radio range.

The remainder of this chapter is as follows. Section 5.2 presents the design of UPL. The algorithm and implementation are discussed in Section 5.3. Section 5.4 shows experimental results and an application example is presented in Section 5.5. Section 5.6 concludes this chapter.

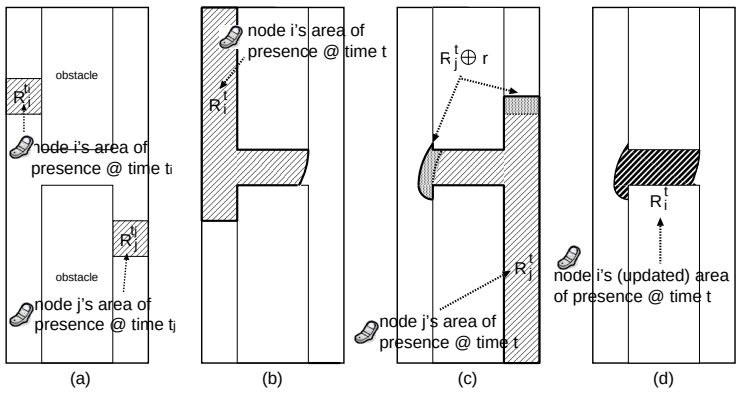


Figure 5.1: Localization process of UPL.

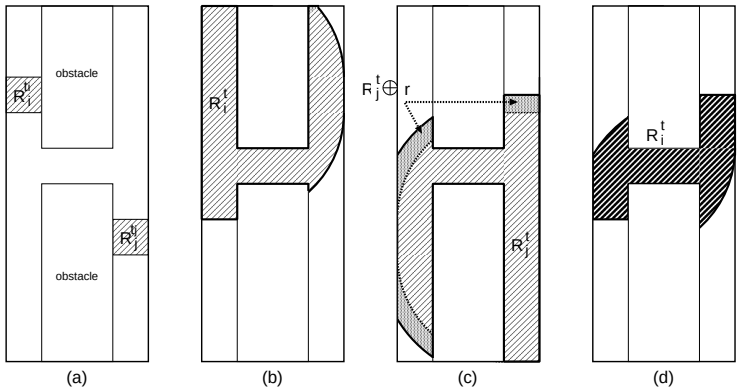


Figure 5.2: Localization process of UPL_{no_obs}.

5.2 Urban Pedestrians Localization: Design and Benefit

5.2.1 Preliminaries

Each node is equipped with a personal area communication device such as ZigBee and Bluetooth. Our algorithm does not depend on specific hardware, however it is not realistic to assume that each portable device continues seeking neighbors which are several tens of meters away, especially in a dense crowd like in downtown, due to battery limitation. Thus as a realistic hardware environment we assume these PAN communication technologies. For simplicity of

Table 5.1: Notations.

Symbol	Notation
r	communication range
V_{max}	maximum speed of nodes
R_i^t	area of presence of node i at time t
$R_i^t \oplus r$	area of presence of node i at time t expanded by communication range r
t_i	time when the last localization was executed at node i
t	current time
Δt_i	elapsed time since t_i, that is $t - t_i$

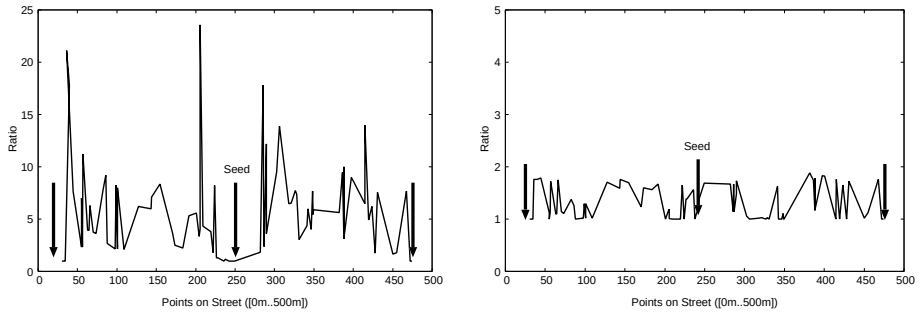
discussion, we assume the same communication range r for all the nodes. We also assume the same *maximum* velocity V_{max} for all the nodes (this does not mean that each node moves with the same speed). Hereafter, we let R_i^t denote an estimated area of presence (or simply called an *area of presence*) of node i at time t. The area of presence of node i is the region in which node i expects to exist.

Each node broadcasts hello messages with regular intervals to its neighbors. A hello message transmitted by node i includes the area of presence $R_i^{t_i}$ and time $\Delta t_i = t - t_i$ where t_i is the time when the last localization was executed at node i and t is the time when the message was transmitted. Thus $R_i^{t_i}$ denotes the most recently updated area of presence of node i, and time duration Δt_i indicates the elapsed time since t_i. Obviously Δt_i is calculated by a local timer. The hello message also includes a common obstacle map M of a target region which is lightweight enough. We assume that this map is initially given by seed nodes, and then it is distributed by hello messages among mobile nodes. The data structures of $R_i^{t_i}$ and M are given later in Section 5.3.1.

5.2.2 Localization Overview

UPL (Urban Pedestrians Localization) algorithm presented in this chapter is a multi-lateration algorithm that updates each node's area of presence when it meets another mobile node (or a seed node) which has also its own area of presence which was updated at a certain time.

When node i receives a hello message from node j, node i immediately runs UPL to update its area of presence. UPL is executed as follows. We denote by



(a) UPL_{no_adhoc} / UPL (b) UPL_{no_obs} / UPL

Figure 5.3: Ratios of average sizes in simplified versions over those in original UPL.

t the time when node i received the hello message. Node i calculates R_i^t and R_j^t from given $R_i^{t_i}$ and $R_j^{t_j}$, respectively. Moreover, based on the calculated R_j^t where node j is expected to exist at time t, node i calculates the region in which node j's signal can be heard at time t. We calculate this region by expanding R_j^t by communication range r and it is denoted by $R_j^t \oplus r$. Finally, we obtain the new area of presence of node i at time t, by intersecting R_i^t and $R_j^t \oplus r$ because node i must be located in both R_i^t and $R_j^t \oplus r$.

We say that R_i^t is *complete* if and only if R_i^t contains the position of node i at time t. Moreover, for two complete areas of presence R_i^t and $\hat{R}_i^t$ of node i at time t, if $|R_i^t| < |\hat{R}_i^t|$ then R_i^t is said to be more *accurate* than $\hat{R}_i^t$. Our goal is to design an algorithm that can determine areas of presence of nodes which are as complete and accurate as possible. Of course prioritizing completeness or accuracy is a trade-off, so in our experiments we pursue this trade-off and discuss better settings.

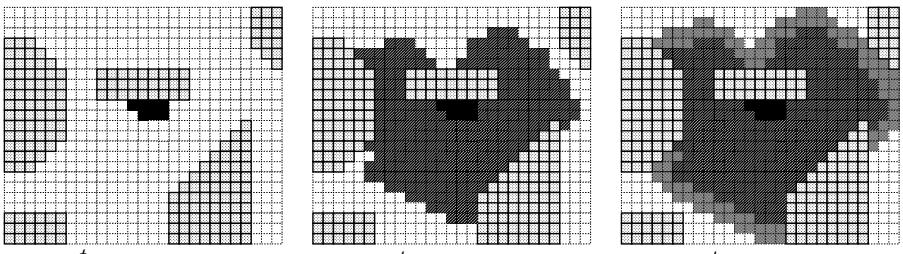
Area of presence information itself can be used for many services. For example, if service providers receive an area of presence of a shopping customer, it may send the information about shops and restaurants in that area. In such a case, to avoid sending useless information for users, precise identification of the area of presence is an important issue. Also, for services which need to identify the location of nodes such as navigation systems, we present a position estimation function that determines the most likelihood point from a given area of presence. This is given in Section 5.3.5.

An example of UPL procedure is illustrated in Fig. 5.1. In Fig. 5.1(a), $R_i^{t_i}$ and $R_j^{t_j}$ are illustrated. Fig. 5.1(b) and Fig. 5.1(c) show R_i^t and R_j^t computed from $R_i^{t_i}$ and $R_j^{t_j}$, respectively. Additionally in Fig. 5.1(c), $R_j^t \oplus r$ is shown. Finally Fig. 5.1(d) shows R_i^t, the intersection of Figs. 5.1(b) and 5.1(c).

As we have seen in the above example, we need to calculate R_i^t and $R_j^t \oplus r$. Again, compared with the existing proposals, the significance of our UPL is two-fold; (i) we consider ad-hoc localization between mobile nodes and (ii) we consider the movement of mobile nodes among obstacles in the algorithm to take into account the practical environments in urban district. To see the effect of (i) and (ii) intuitively, we have measured the sizes of the areas of presence in two simplified versions; UPL_{no_adhoc} which does not perform ad-hoc localization between mobile nodes, and UPL_{no_obs} which does not utilize obstacle information to predict the movement of nodes. The simulations were conducted in a $500m \times 500m$ region where 16 seeds and 2,000 mobile nodes were deployed on 8 vertical and horizontal streets (for the other settings, see Section 5.4). We focused on one of the streets, and plotted on each point on the street the ratio of the average size of the areas of presence of nodes at the point in UPL_{no_adhoc} over that in the original UPL in Fig. 5.3(a) and the ratio of the average size in UPL_{no_obs} over that in the original UPL in Fig. 5.3(b). From Fig. 5.3(a), if we do not perform ad-hoc localization, the average areas of presence are much larger than those in UPL at some points. On the other hand, from Fig. 5.3(b), if we do not use obstacle information, the average areas of presence are about twice as large as those of UPL in the worst case and almost nodes are uniformly affected.

Moreover, to see how the consideration of obstacles is effective to reduce the area of presence in more details, we present the localization process by UPL_{no_obs} in Fig. 5.2, which corresponds to that of UPL in Fig. 5.1. Clearly, the calculated areas of presence, R_i^t and R_j^t unnecessarily cover the region beyond the obstacles, and as a result, the obtained R_i^t in Fig. 5.2(d) is much larger than that in Fig. 5.1(d).

To take benefit from ad-hoc localization by mobile nodes and obstacle information, we need to calculate the movement of nodes among obstacles. Thus we carefully design the data structures and the algorithms which do not overload small hand-held devices with poor memory size and poor computational capa-



(a) $R_j^{t_j}$ (black grids) and obstacles (lightly shaded grids with solid lines) (grid size $w = 1$).

(b) R_j^t (black and dark grids; $V_{max} \cdot \Delta t_j = 10$; $\Delta t_j = t - t_j$)).

(c) $R_j^t \oplus r$ (black, dark and gray grids; $r = 2$).

Figure 5.4: Computing area of presence and its expansion by communication range.

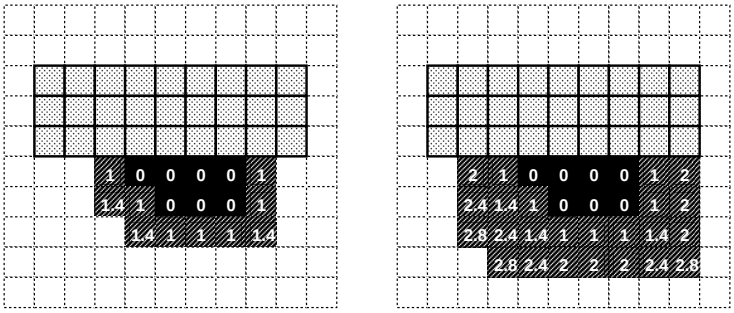
bility. In the next section we present the data structure and algorithm followed by discussion about implementation issues.

5.3 Data Structure, Algorithm and Implementation Issues

5.3.1 Area of Presence and Obstacle Map

Several data structures have been introduced and used to represent an area of presence in past localization algorithms. In Ref. [24], some data structures are given and discussed in details. In the most simplest way, we may use circles or simple polygons such as rectangles to approximate areas of presence, but obviously it lacks the accuracy of approximation. MCL [16] uses a set of randomly selected points to represent an area of presence, where simplicity is a merit for small hand-held devices. Sextant [24] utilizes a list of representative points and the area is approximated as a set of Bezier curves between those points.

Unlike these methods, we divide a target region into small grids and represent areas of presence and obstacles by sets of grids. Here, the form of each area of presence in UPL is more complex than existing methods because we consider regions restricted by obstacles and the movement of nodes in such complex regions. There are some methods that deal with obstacles. For example, Sextant [24] takes into account an area restricted by obstacles. However, it does not need to consider the computation of movement of nodes because it deals with



(a) After the 1st step. (b) After the 2nd step.

Figure 5.5: 1st and 2nd steps of APC algorithm (value inside a grid represents distance from $R_j^{t_j}$).

stationary nodes. Considering the fact that in our case the operations on areas of presence are more complex, we should benefit from simple data structure. Instead, data size may be large. Later we will analyze how much memory is required to execute the algorithm.

5.3.2 UPL Algorithm Description

Computing Area of Presence

We present an algorithm to compute R_j^t, an area of presence of node j at time t, from given $R_j^{t_j}$ $(t_j < t)$, $\Delta t_j = t - t_j$ and an obstacle map M. R_j^t and M are represented by sets of grids. This algorithm is referred to as APC (Area of Presence Computation) algorithm and does not require complex functions like ones used in graphics libraries, instead we only use simple operations which are lightweight enough.

First, we represent the map M by set FS, the set of all the grids in the movable space (free space) of M. That is, FS is the set of all the grids which are not included by the obstacles. Basically, to calculate R_j^t, APC algorithm adds to $R_j^{t_j}$ the grids in FS within $V_{max} \cdot \Delta t_j$ distance from $R_j^{t_j}$. However, it may be expensive under the existence of obstacles to calculate the shortest distance from $R_j^{t_j}$ for each grid in FS. Therefore, we design the algorithm such that we can expand $R_j^{t_j}$ step by step while keeping a certain accuracy.

We introduce some terminologies and notations. For grid g, a grid which shares a side with g is called a *side grid*, and a grid which is not a side grid of

```
1  R_j^t = R_j^{t_j}; B_0 =set of border grids of R_j^{t_j};
2  FS =set of grids in free space except R_j^{t_j}; h = 0;
3  do {
4     B_{h+1} = ∅;
5     for each g ∈ B_h {
6        for each side grid g' of g in FS {
7           if (d(g) + w ≤ V_max · Δt_j) then {
8              B_{h+1} = B_{h+1} ∪ {g'}; d(g') = d(g) + w;
9              FS = FS − g';
10          }
11       }
12       for each diagonal grid g'' of g in FS {
13          if (d(g) + √2 w ≤ V_max · Δt_j) then {
14             B_{h+1} = B_{h+1} ∪ {g''}; d(g'') = d(g) + √2 w;
15             FS = FS − g'';
16          }
17       }
18    }
19    R_j^t = R_j^t ∪ B_{h+1}; h = h + 1;
20 } while (B_h ≠ ∅)
```

Figure 5.6: Area of presence computation (APC) algorithm.

g and shares a single vertex with g is called a *diagonal grid*. The side grids and diagonal grids are called *neighboring grids*. Also, for a set G of grids, a grid in G which has at least one neighboring grid outside G is called a *border grid*. For each grid g, we let $d(g)$ denote the shortest distance from $R_j^{t_j}$ ($d(g) = 0$ if g is in $R_j^{t_j}$). Here, the distance of two neighboring grids is the Euclid distance between the centers of the grids.

APC algorithm is described as follows. We start with $R_j^t = R_j^{t_j}$ and iterate the following procedure. For each border grid g of R_j^t with the shortest distance $d(g)$, we add to R_j^t each side grid g' of g in FS if $d(g) + w \leq V_{max} \cdot \Delta t_j$ where w is the side length of a grid. If g' is added, we set the shortest distance of g' as $d(g') = d(g) + w$. Similarly, we add to R_j^t each diagonal grid g'' of g in FS if $d(g'') + \sqrt{2}w \leq V_{max} \cdot \Delta t_j$. If g'' is added, we set the shortest distance of g'' as $d(g'') = d(g) + \sqrt{2}w$. Then we remove all g' and g'' which have been added to R_j^t from FS. This procedure is repeated until there are no new grids in FS which can be added to R_j^t.

Fig. 5.4 shows examples of $R_j^{t_j}$, R_j^t and $R_j^t \oplus r$. Also in Fig. 5.5 we illustrate how $R_j^{t_j}$ expands at each step of the algorithm where $\sqrt{2} \approx 1.4$. The formal

description of this APC algorithm is given in Fig. 5.6.

Expanding Area of Presence by Communication Range

Then we compute $R_j^t \oplus r$, the area of presence of node j at time t expanded by communication range r.

For this purpose, we may apply APC algorithm described in Section 5.3.2. Here, radio propagation may be affected by phenomena such as fading and diffraction especially in city sections. Therefore, ideally, we should design a propagation prediction algorithm considering these phenomena. However, in our environment, it is not reasonable to implement such a complex algorithm. Also, as we stated briefly in Section 5.2.1, we assume PAN communication devices with relatively small communication range. Therefore, we may ignore the impact by those phenomena and we simply use APC algorithm to expand R_j^t.

Intersecting Two Areas

Intersecting R_i^t and $R_j^t \oplus r$ is simple. We seek grids which are included in the two areas. Finally we obtain the new area of presence, $R_i^t \cap \{R_j^t \oplus r\}$.

5.3.3 Memory Space and Computation Complexity Issues

We discuss implementation issues of the APC (Area of Presence Computation) algorithm. In particular, we look at physical memory usage to show that we can implement UPL for small hand-held devices.

We first analyze $|R_j^t \oplus r|$, the size of the computed area of presence expanded by communication range. We consider the worst case where no obstacle exists. We let dia_{max} denote the maximum diameter of $R_j^{t_j}$ just after the execution of the last localization at node i at time t_j and Δt_{max} denote the maximum value of Δt_j. Then,

$$|R_j^t \oplus r| \leq \pi(\frac{dia_{max}}{2} + V_{max} \cdot \Delta t_{max} + r)^2$$

We assume that node i receives at most n nodes' areas of presence at time t. Node i needs space to store at most

$$(n+1)\frac{\pi(\frac{dia_{max}}{2} + V_{max} \cdot \Delta t_{max} + r)^2}{w^2}$$

grids for the areas of presence, and

$$\frac{FS}{w^2}$$

grids for an obstacle map where FS denotes the free space in the obstacle map. For each grid in an area of presence, vertical and horizontal offsets and the shortest path distance of the grid need to be stored. Here, we borrow some parameter settings from our experiments of Section 5.4 to see the memory usage in practical situation. The target field size is $500m \times 500m$ and the grid size is $2m \times 2m$. Then we have 250×250 grids in the field and 1 byte is required to represent an offset in [0..250). We have set $V_{max} = 2m/s$ and $r = 10m$. Also to represent a shortest distance, 1 byte is enough. Thus for each grid in an area of presence we need 3 bytes, and for each grid in FS we need 2 bytes. From the experimental results, the area of presence was at most $200m^2$ and $dia_{max} = 25m$ if the street width is $8m$. Thus we set $\frac{dia_{max}}{2} = 12.5m$. Also, from the experimental results $\Delta t_{max} = 2sec.$ and $n = 1$ in most cases. As a result, each node needs to have at most 1.1k grids in the areas of presence where each grid requires 3 bytes, and $62.5k$ grids in FS where each grid requires 2 bytes. Thus around 130kbyte memory is required. This is reasonable enough for recent mobile terminals which have memory in mega-byte order, and manageable even by small sensor nodes.

As a computation complexity, UPL does not require complex operations like those used in graphics library, but only requires iteration of simple operations like list search. We are now going to implement our UPL algorithm in MOTE to show that it is executable by small devices.

5.3.4 Other Optimization

The existing techniques have utilized some other information to assist more accurate localization. Some techniques maintain and use the history of signal receptions from a seed. Concretely, by knowing the timing when the node entered or left the communication range of a seed, we may be able to identify the position of the node exactly on the disc edge of the communication range of the seed. Also some try to extend the communication range of a seed by multi-hop propagation of the seed information. In our experiments, we have incorporated these techniques for more accurate localization and position estimation. The detailed implementation is explained in Section 5.4.

Table 5.2: Simulation parameters.

Region size	500m×500m
Grid length (w)	1, **2** or 4 (m)
Radio range (r)	10 (m)
# of Nodes	500, 1,000, **2,000** or 3,000
# of Seeds	8, 12 or **16**
Initial Node deployment	Uniform distribution
Mobility model	Random Street Decision Model
Speed distribution	Uniform distribution of **[1.0, 2.0]**, $[3.0, 4.0]$, $[5.0, 6.0]$, $[7.0, 8.0]$ or $[9.0, 10.0]$ (m/s)
Hello message interval	1, **2**, 5 or 10 (s)

5.3.5 Position Estimation

Here, in case that we need to identify the location of a node from its area of presence, we give a position estimation function that determines the most likelihood point in the area of presence. We have used the following estimation function to determine the most likelihood point p from an area of presence R and obstacle map M;

$$\text{select } p \in R \text{ that minimizes } \max_{p' \in R} dist(p, p') \tag{5.1}$$

where $dist(p, p')$ is the shortest distance between p and p' on M (that is, the shortest distance among obstacles). Considering the fact that the actual point exists within R in most cases (later we will show that most areas of presence are complete in UPL), selecting such p that minimizes the maximum distance between p and another point p' in R is helpful to minimize errors of positioning.

5.4 Performance Evaluation

We have evaluated the performance of UPL through our network simulator *MobiREAL* [25, 26, 27], which has been developed based on *GTNetS* (Georgia Tech. Network Simulator [96]) and can create realistic behavior of mobile nodes under given city maps.

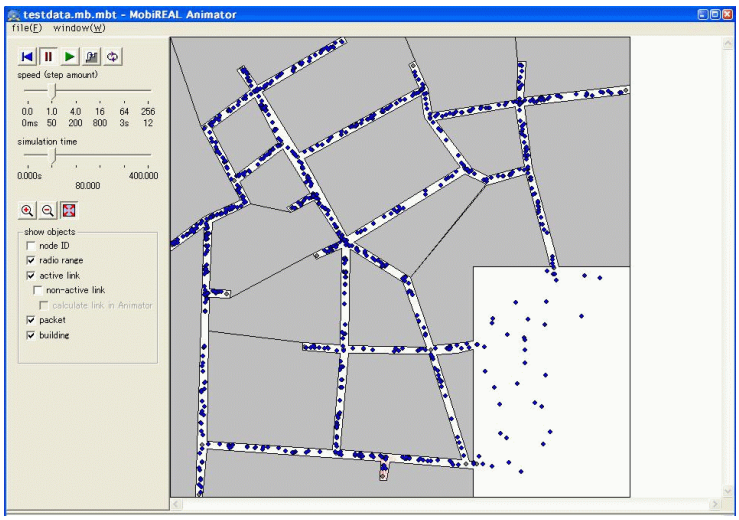


Figure 5.7: Osaka downtown region (snapshot from MobiREAL animator).

5.4.1 Simulation Settings

We have used two maps, (i) *Manhattan region* [101] of 500m×500m with 8 streets and (ii) the real map of 500m×500m region in front of the Osaka train station (Fig. 5.7) called *Osaka downtown region*. In the both regions, nodes were uniformly deployed initially, and then they are moved according to "random-street-decision" mobility where at each intersection each node decides to which direction it goes, except the backward direction. The other settings are described in Table 5.2 where default values are emphasized by bold font.

As we stated in Section 5.3.4, some techniques for more precise estimation of positions have been presented in some literatures. Especially, MCL [16] provides two effective techniques, one of which uses the history of signal receptions from a seed (this history information is called leaver and arriver information), and another uses two-hop advertisement of seed positions to extend the coverage of seeds. In the two-hops mode, nodes which have received the position information from seeds immediately forward the information to their neighbors. In our experiments, we have borrowed these sophisticated ideas and implemented those techniques. Due to space limitations we omit the implementation details but interested readers may refer to Ref. [16].

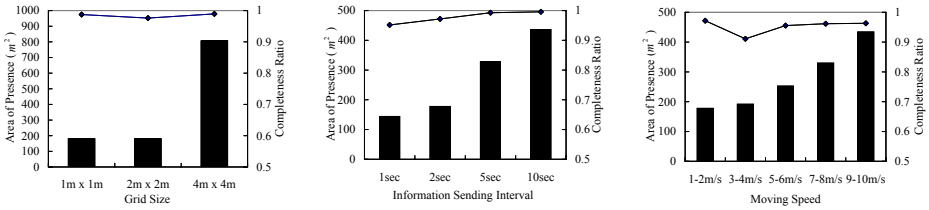
We conducted three types of experiments. The experiments of the first type were conducted to see the performance characteristics of UPL varying several parameters. In particular, we pursue the trade-off between completeness (whether an area of presence includes the current position of a node or not) and accuracy (the size of the area of presence), and show that our UPL could achieve a reasonable trade-off. The experiments of the second type examine the effects of (i) the ad-hoc localization and (ii) precise calculation of node movement among obstacles. For this purpose, we also examine the performance of two simplified versions of UPL, UPL_{no_adhoc} and UPL_{no_obs}, part of the results have already shown in Section 5.2.2 and Fig. 5.3. In these experiments we have used the Manhattan region. Finally, the experiments of the third type were conducted to see the performance compared with typical existing methods. We have selected Amorphous [20] which performs multi-hop cooperative multi-lateration, and MCL [16] which uses short-range seeds. These methods estimate the position of each node, therefore, we have used the position estimation function (5.1) of Section 5.3.5. In these experiments we have used Osaka downtown region to see the practical performance in a real region.

The experimental results of these three types are presented and analyzed in the following sections, Sections 5.4.2, 5.4.3 and 5.4.4, respectively.

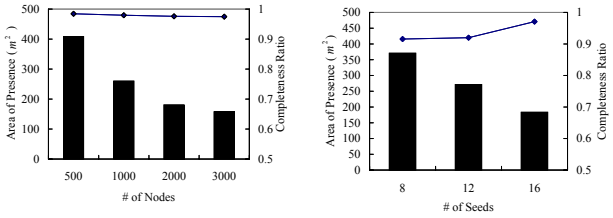
5.4.2 Impact of Parameters and Environments on Performance

We have conducted the simulations of the first type in the Manhattan region. For the areas of presence of nodes, we have measured their accuracy and the completeness. To see the accuracy, we have measured the sizes of areas of presence. For completeness, we have measured the completeness ratio, which is the ratio of the complete areas of presence among all the examined areas of presence. The average sizes and ratios are derived from 20 simulation cases.

Grid Size The grid length w may affect both accuracy and completeness. We have evaluated the average sizes and completeness ratios of areas of presence under different values of grid length w ($1m$, $2m$ and $4m$). The result is shown in Fig. 5.8(a). From the result, we do not see the difference of completeness ratios for different values of w (they are close to 1.0, the perfect completeness),



(a) vs. Grid size (b) vs. Hello message interval (c) vs. Moving speed



(d) vs. # of nodes (e) vs. # of seeds

Figure 5.8: Impact of parameters and environments on performance.

and we can see explicit difference of the sizes. This is because smaller grids can represent the borders of areas of presence more accurately. In the case of $w = 4$, the average size is $800m^2$, which is rather large compared with the cases of $w = 2$ and $w = 1$. Also we do not see big difference between the cases of $w = 1$ and $w = 2$. Considering the fact that the grid size quadratically affects memory space as we discussed in Section 5.3.3, we may select $w = 2$.

Hello Message Interval The frequency of hello message transmissions will affect the accuracy because receiving "fresh" area of presence information is much more useful to reduce the area of presence for the receiver. In order for a node (say node i) to receive a hello message from a node (say node j) which is moving toward node i, the interval must not be greater than $\frac{r}{V_{max}}$. In this experiment $r = 10m$ and $V_{max} = 2$, thus 5 seconds are the maximum interval. We see the accuracy in different intervals (1, 2, 5 and 10 seconds). The result is shown in Fig. 5.8(b). From the result, 2 seconds seem the most reasonable interval because it achieved similar size as the case of 1 second interval with the less number of messages.

Moving Speed As movement speeds become large, there is a merit for each node that it may meet more nodes. However quick expansion of areas of presence is a demerit. To see what happens if we increase the speeds of nodes, we have varied the minimum and maximum speeds. The result is shown in Fig. 5.8(c). The size of the area of presence increases linearly as the speed increases. Thus slow speed is better for accuracy. Since increasing speeds has similar effect with increasing hello message intervals, this is the result as expected. This result proves our localization is well fit for slow pedestrians.

Number of Nodes Obviously, UPL is largely affected by node density. Thus we have varied the number of nodes and have measured the accuracy and completeness. Fig. 5.8(d) shows the result. From the result, we need a certain density to achieve accurate areas of presence. However, this means that we need to *encounter* with a certain number of nodes, and does *not* mean that a dense ad-hoc network is required. Such feature is good in city sections where it is hard to assume that large ad-hoc networks are constructed and maintained.

Number of Seeds One of the advantages of UPL is that it does not require many seeds. To see what happens if we decrease the number of seeds, we have varied the number of seeds (8, 12 and 16). The result is shown in Fig. 5.8(e). This is the natural result where the accuracy increases as the number of seeds increases. Here, 16 seeds with 10m transmission range only cover 2% of the $25 \times 10^4 m^2$ region. Even in such a region with sparsely deployed seeds, UPL could achieve reasonable accuracy where $200m^2$ indicates that we can identify a node on a street with 8m width within 25m length.

Totally, UPL could achieve enough accuracy (around $200m^2$) with almost perfect completeness, by appropriate parameter settings and environments.

5.4.3 Effect of Ad-hoc Localization and Precise Calculation of Movement

To see the effectiveness of our two key ideas, (i) ad-hoc localization between mobile nodes, and (ii) precise calculation of nodes' movement among obstacles, we revisit the result in Fig. 5.3 and show the distributions of the ratios of sizes at the points in the whole region in Fig. 5.9. Their average values are described in Table 5.3 as well as the average completeness ratios. The default parameter

Table 5.3: Average ratios of sizes and completeness ratios.

	Ratio of sizes	Completeness ratio
UPL_{no_adhoc}	5.293	0.981
UPL_{no_obs}	1.233	0.983
UPL	1.000	0.983

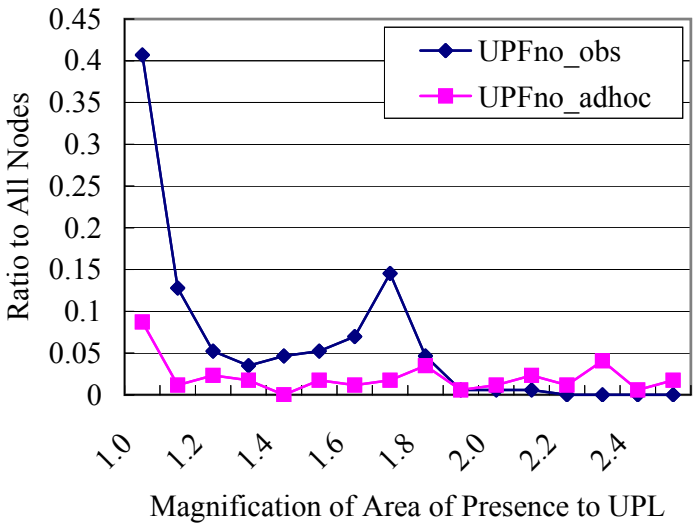


Figure 5.9: Distribution of ratios of sizes of areas of presence in UPL_{no_adhoc} and UPL_{no_obs} over those in UPL.

settings in Table 5.2 were used in the experiments. From Fig. 5.9, obstacle information gives a certain amount of impact on accuracy of many nodes. On the other hand, ad-hoc localization dramatically improves the accuracy of some nodes. This fact indicates that in some places which are away from seeds, ad-hoc localization is very helpful. From Table 5.3, obstacle information and ad-hoc localization reduce 19% and 81% of each area of presence in average, respectively. Overall, we can say that ad-hoc localization dramatically improves the accuracy of some nodes, and obstacle information well helps further improvement uniformly.

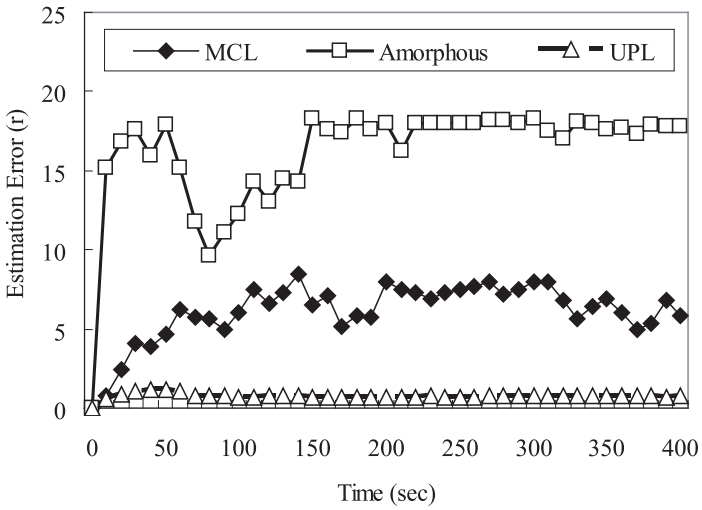


Figure 5.10: Average estimated position errors.

5.4.4 Comparison with Other Approaches

For comparison purpose, we have measured the estimated position errors of UPL, MCL[16] and Amorphous[20] using Osaka downtown region. The default parameters in Table 5.2 were used in the experiment. Fig. 5.10 shows the position errors (average of 10 simulations) regarding the communication range ($r = 10m$) according to the progress of simulation time. Here, the important fact is that Amorphous achieved relatively low accuracy, and MCL performed better than Amorphous even though MCL does not use ad-hoc networks and just uses direct information from seeds. This is because accurate estimation by a hop-based technique may be difficult in urban district where most space is restricted by obstacles. On the other hand, UPL outperforms these methods and the error is at most $0.8r$, that is, $8m$. This is reasonably small error for services like navigation in cities, identification of nearby shops and restaurants, or tracking of children.

```
GreedyMode(i, d)
 1  mindist = dist(l_i, l_d);
 2  minid = i;
 3  foreach(n ∈ NeighborList){
 4     if( dist(l_n, l_d) < mindist ){
 5        mindist = dist(l_n, l_d);
 6        minid = n;}
 7  } //Decides the next hop greedily
 8  if( minid != i)//forward to the next hop
 9     SendTo(minid);
10  else //cannot get closer
11     PerimeterMode(i, d);
```

Figure 5.11: Forwarding algorithm of GPSR.

```
Planarize(i)
 1  foreach(n ∈ NeighborList){
 2     foreach(w ∈ NeighborList){
 3        if(w == n) continue;
 4        else if(dist(l_i, l_n)
 5             > max[dist(l_i, l_w), dist(l_n, l_w)]){
 6           eliminate edge(i, n);
 7           break;
 8        }
 9     }
10  }
```

Figure 5.12: Relative neighborhood graph generation algorithm.

5.5 Application Example: GPSR Using UPL

5.5.1 Overview of GPSR

GPSR assumes that each node knows its own accurate position. Hereafter, l_i denotes the position of node i. In order to know the positions of the destinations, some location services such as MLS [102] may be used.

Each node maintains its *neighbor table* where each entry consists of the identification and position information of a neighbor. The neighbor table is updated by beacons periodically broadcast by neighbors. Fig. 5.11 shows the forwarding algorithm of GPSR. When node i forwards a message destined for node d, i forwards the message to the neighbor which is geographically closer to d than i and the closest in all the neighbors included in i's neighbor table. This

Table 5.4: Simulation parameters.

Radio range (r)	10 (m)
Bandwidth	1($Mbps$)
Beacon interval	1 (s)
Moving speed	Randomly selected from [0.1, 1.0] (m/s)

forwarding policy is called the *greedy mode*. In addition, the *perimeter mode* is used to forward messages when nodes cannot find closer neighbors in the greedy mode.

In the perimeter mode, planar graphs are locally constructed based on location information. On the planar graphs, messages are delivered on the faces, which are the polygons on the baseline between the source and th e destination, following the right-hand rule. Consequently, messages are delivered whenever the source and the destination are connected because the planar graphs do not have crossing edges. The planar graph construction algorithm is described in Fig. 5.12. In this algorithm, a Relative Neighborhood Graph (RNG) is locally constructed using neighbors' location information.

5.5.2 Design of GPSR Using UPL

In GPSR, each node periodically broadcasts a beacon including its location information to its neighbors. On the other hand, UPL also uses beacons that contain senders' areas of presence. Thus, we can design GPSR by using beacons of UPL because the position of node j can be calculated from j's area of presence.

The following is a detailed description of the process.

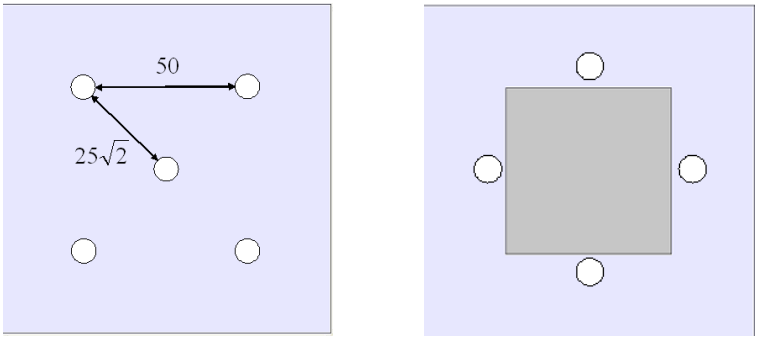
1. Node i runs UPL and updates R_i by the intersection of R_i and $R_j \oplus r$ when i receives a beacon from j.

2. Node i estimates position of j based on R_j by the position estimation function (5.1) of Section 5.3.5.

5.5.3 Performance Evaluation

Simulation Settings

We have evaluated GPSR using UPL by MobiREAL simulator. Hereafter, we denote GPSR using UPL by UPL for simplicity of notation. In the simulation,



(a) Free space. (b) Obstacle map.

Figure 5.13: Simulation map.

the followings are compared with UPL.

IDEAL In this case, GPSR is running in the ideal environment. Each node always knows its own and neighbors' exact location information without any beacons. This setting derives the best performance of GPSR.

GPS Each node is equipped with GPS and broadcasts a beacon including the node's exact location information periodically. GPSR was executed in this environment. In the experiments, beacon intervals were set equal to that of UPL.

Simulation parameters are shown in Table 5.4. In the experiments, the number of nodes was changed. We have used two simulation maps: a free space of 100m×100m shown in Fig. 5.13(a) and an obstacle map where a square obstacle of 50m×50m is located at the center (see Fig. 5.13(b)). In the both maps, base stations that advertise their accurate positions are located at the circles shown in Fig. 5.13(a) and Fig. 5.13(b). We used the random walk mobility model [103] for the movement of mobile nodes and the maximum moving speed V_{max} used in UPL was set to $1.0(m/s)$.

On the above settings, we have measured packet delivery ratios and message delay. Each base station randomly selects another base station in every second and sends a 512 bytes message to it.

Results

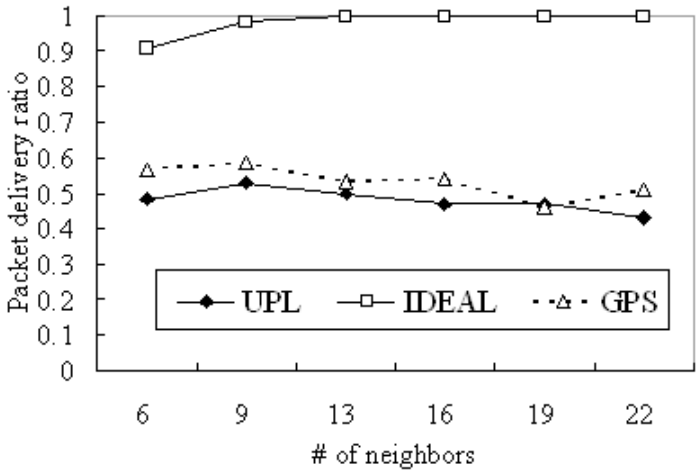


Figure 5.14: Packet delivery ratios in the free space.

Performance in Free Space Fig. 5.14 shows packet delivery ratios in the free space. The packet delivery ratios in GPS and UPL decrease approximately 50% compared to IDEAL. The reason is that in GPS and UPL, there are some location errors in neighbor tables due to the movements of nodes.

One reason why the position errors affect the performance of GPSR is that neighbors included in a neighbor table have sometimes already moved out of the radio range when a node tries to forward a message. In this case, messages cannot be forwarded. Especially in the greedy mode, a forwarding node may often choose a neighbor near the boundary of the radio range since it is probably the closest to the destination. This decreases the packet delivery ratios in GPS and UPL. In order to avoid such situations, Ref. [104] proposes a routing protocol which uses the velocity vectors of neighbors.

From the results shown in Fig. 5.14 and Fig. 5.15, we can see that UPL achieves similar performance to GPS and therefore could confirm the effectiveness of the ad-hoc localization.

Effects of Obstacles Fig. 5.16 and Fig. 5.17 respectively show the packet delivery ratios and message delay in the obstacle map. In Fig. 5.16, the packet delivery ratio of UPL is less than that of GPS. This is because planar graphs

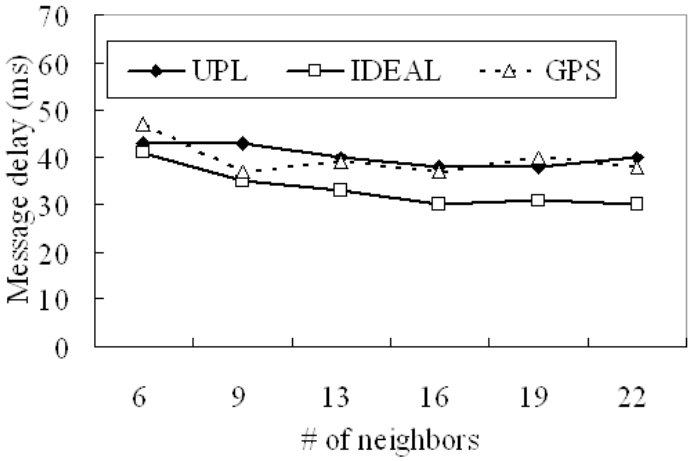


Figure 5.15: Message delay in the free space.

may be wrong when the position information contains large error. Consequently, loops and network partitions may occur. Nevertheless, UPL is comparable to GPS in high node density because of decreased position error.

Position Estimation Error The average position estimation errors in the free space and in the obstacle map are shown in Fig. 5.18 and Fig. 5.19, respectively. In the free space, advertisements from the base stations are easily propagated through ad-hoc communication around the center of the region. In addition, nodes in the free space receive advertisements from the base stations directly more often than in the obstacle map. These are the reasons for the smaller position estimation errors in the free space.

From the results shown in Fig. 5.14–5.19, we can see that UPL achieves comparable performance to GPS when the position estimation error is less than 10m where the wireless communication range is 10m.

5.6 Conclusion

In this chapter, we have proposed a range-free localization algorithm called UPL (Urban Pedestrians Localization) for positioning mobile users in urban districts. In UPL, we assume that each mobile node cannot expect to meet location seeds

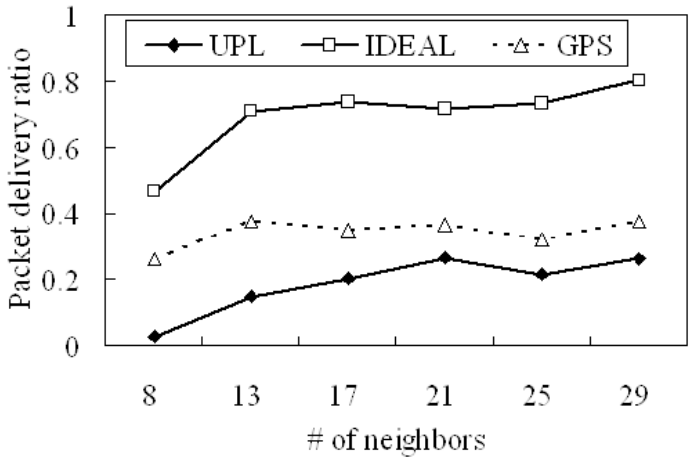


Figure 5.16: Packet delivery ratios in the obstacle map.

frequently. Therefore, each mobile node uses the location information from neighboring nodes, which are also mobile. In such a situation, it is important for highly accurate positioning to predict in which area a node exists after a certain time. For such a purpose, we employ obstacle information to precisely determine the movable area of nodes between obstacles. Simulation results have shown that the positioning error is 8m where radio range is 10m.

We have applied UPL to GPSR and evaluated it in order to see the effectiveness of UPL when used in geographic routing protocols. From the results, we have confirmed UPL could help GPSR to be GPS-free, without excessively sacrificing its performance.

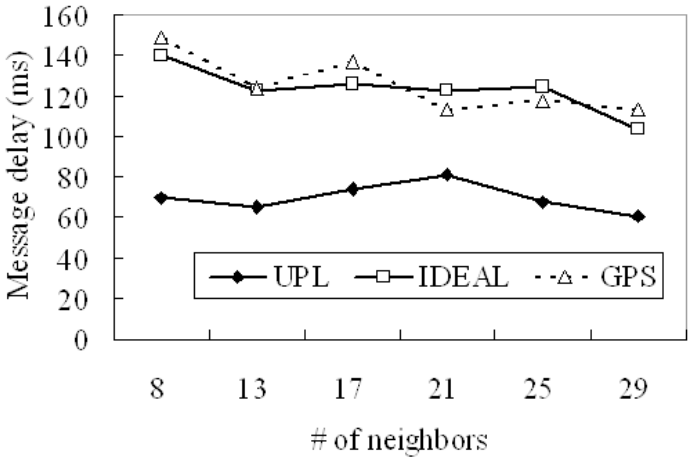


Figure 5.17: Message delay in the obstacle map.

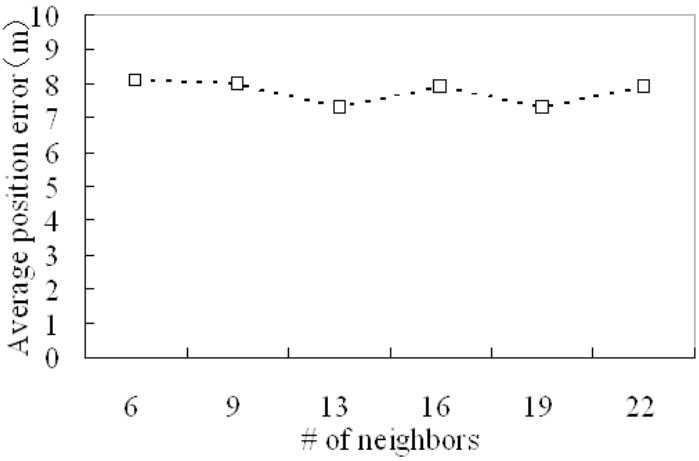


Figure 5.18: Average position estimation error in the free space.

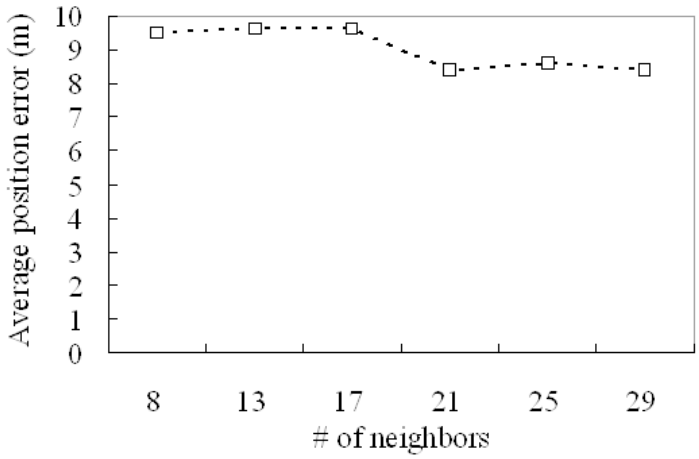


Figure 5.19: Average position estimation error in the obstacle map.

Chapter 6

Conclusion

In this thesis, the following two research topics have been studied to increase reliability of networks: (1) a cooperative network monitoring technique with high availability to reduce workloads of network administrators, and (2) a method for ad-hoc topology formation to achieve stable communication on MANETs.

The existing work related with those topics was summarized in Chapter 2. First, some related work concerned with network management was mentioned. Next, we referred to some mobility models for simulations. Techniques for stable ad-hoc communication such as geographic routing protocols were also discussed. Finally, various localization algorithms were surveyed.

In Chapter 3, a cooperative network monitoring technique has been proposed. The proposed technique aims at reducing workloads of network administrators and detecting damaged areas quickly and thoroughly when problems such as infection of viruses and DoS/DDoS attacks occur. For this purpose, we have proposed a middleware for efficient and robust network monitoring based on autonomous group formation. Network administrators can easily describe network management scenarios by using the APIs provided by the middleware. When a problem occurs in multiple segments, the corresponding monitoring nodes form a group dynamically using logical neighborhood relations. By communication among group members, they can share the information about the monitoring items. Based on the pre-defined management scenario, the nodes in the group carry out the reactive actions autonomously. This feature increases the availability of the proposed technique.

Next, we have analyzed the effect of mobility patterns on MANET protocols

in the Chapter 4. The simulation results have shown the importance of mobility consideration for the performance of MANET protocols. As a case study, the Connectivity-based Direction Estimation (CDE) was proposed to estimate neighbors' moving direction by monitoring beacons. We have applied CDE to the routing protocol DSR and evaluated its performance through simulation. The results show that the stability of ad-hoc communication has been improved by considering the movements of nodes.

Finally, in order to increase the stability and reliability of ad-hoc communication, Chapter 5 presented a range-free localization algorithm called UPL (Urban Pedestrians Localization) for positioning mobile users in urban districts. In UPL, we do not assume that each mobile node can meet location seeds frequently. Therefore, each mobile node uses the location information received from its neighboring nodes, which are also mobile. We employ obstacle information to precisely determine the movable areas of nodes between obstacles, and we use the information to increase the accuracy of positions of mobile nodes. We have confirmed the effectiveness of UPL through simulation.

Acknowledgement

I would like to express my sincerely appreciation to continued support of my supervisor Professor Teruo Higashino of Osaka University through trials and tribulations of this Ph.D thesis. I would like also thank his encouragement and invaluable comments in preparing this thesis.

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 am also very grateful to Professor Akio Nakata of Hiroshima City University, Associate Professor Hirozumi Yamaguchi of Osaka University and Associate Professor Keiichi Yasumoto of Nara Institute of Science and Technology who have provided many valuable comments and discussions.

I would like to thank Assistant Professor Takaaki Umedu of Osaka University for his valuable comments.

Thanks go to all the members of Higashino Laboratory of Osaka University for their helpful advice.

Finally, I would like to thank my family, especially my parents, for their help and understanding.

Bibliography

[1] C. Toh, *Ad hoc mobile wireless networks - protocols and systems* -, Pearson Education, 2002.

[2] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *Mobile ad hoc networking*, IEEE Press, 2004.

[3] T. Higashino, H. Yamaguchi, K. Yasumoto, and K. Shimada, "Toward future innovation of mobile communication systems – current requirements for research –", in *Proceedings of International Conference on Mobile Computing and Ubiquitous Networking (ICMU2005)*, 2005, pp. 2–7, (invited paper).

[4] T. Higashino and H. Yamaguchi, "A testing architecture for designing high-reliable manet protocols", in *Proceedings of IFIP International Conference on Formal Techniques for Networked and Distributed Systems (LNCS3731)*, 2005, pp. 20–23, (invited paper).

[5] L. Garber, "Denial-of-service attacks rip the Internet", *IEEE Computer*, vol. 33, no. 4, pp. 12–17, 2000.

[6] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP", in *Proceedings of IEEE Symposium on Security and Privacy*, 1997, pp. 208–223.

[7] D. Moore, G. Voelker, and S. Savage, "Inferring internet denial-of-service activity", in *Proceedings of USENIX Security Symposium*, 2001, pp. 9–22.

[8] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald, "Detecting SPIT calls by checking human communication pat-

terns", in *Proceedings of IEEE International Conference on Communications (ICC)*, 2007, pp. 1979–1984.

[9] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1998, pp. 85–97.

[10] C. Bettstetter, "Mobility modeling in wireless networks: Categorization, smooth movement, and border effects", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 3, pp. 55–67, 2001.

[11] J. Yoon, M. Liu, and B. Noble, "Sound mobility models", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2003, pp. 205–216.

[12] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1999, pp. 195–206.

[13] X. Hong, T. Kwon, M. Gerla, D. Gu, and G. Pei, "A mobility framework for ad hoc wireless networks", in *Proceedings of IEEE International Conference on Mobile Data Management (MDM)*, 2001, pp. 185–196.

[14] F. Bai, N. Sadagopan, and A. Helmy, "The IMPORTANT framework for analyzing the impact of mobility on performance of routing for ad hoc networks", *AdHoc Networks Journal*, vol. 1, no. 4, pp. 383–403, 2003.

[15] D. Choffnes and F. Bustamante, "An integrated mobility and traffic model for vehicular wireless networks", in *Proceedings of ACM Workshop on Vehicular Ad Hoc Networks (VANET)*, 2005, pp. 69–78.

[16] L. Hu and D. Evans, "Localization for mobile sensor networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004, pp. 45–57.

[17] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)", in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, 2001, pp. 2926–2931.

[18] D. Niculescu and B. Nath, "DV-based positioning in ad hoc networks", *Journal of Telecommunication Systems*, vol. 22, no. 1–4, pp. 267–280, 2003.

[19] C. Savarese and J. Rabaey, "Robust positioning algorithms for distributed ad hoc wireless sensor networks", in *Proceedings of USENIX Annual Technical Conference*, 2002, pp. 317–328.

[20] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network", in *Proceedings of IEEE/ACM Information Processing in Sensor Networks (IPSN)*, 2003, pp. 333–348.

[21] H. Wu, C. Wang, and N. Tzeng, "Novel self-configurable positioning technique for multihop wireless networks", *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 609–621, 2005.

[22] J. Albowicz, A. Chen, and L. Zhang, "Recursive position estimation in sensor networks", in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2001, pp. 35–41.

[23] R. Bischoff and R. Wattenhofer, "Analyzing connectivity-based multihop ad-hoc positioning", in *Proceedings of Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2004, p. 165.

[24] S. Guha, R. Murty, and E. Sirer, "Sextant: a unified node and event localization framework using non-convex constraints", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005, pp. 205–216.

[25] K. Maeda, K. Sato, K. Konishi, A. Yamasaki, A. Uchiyama, H. Yamaguchi, K. Yasumoto, and T. Higashino, "Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network

simulation", in *Proceedings of the ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2005, pp. 151–158.

[26] K. Konishi, K. Maeda, K. Sato, A. Yamasaki, H. Yamaguchi, K. Yasumoto, and T. Higashino, "MobiREAL simulator – evaluating MANET applications in real environments –", in *Proceedings of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2005, pp. 499–502.

[27] "MobiREAL web page", `http://www.mobireal.net`.

[28] "MRTG: The multi router traffic grapher", `http://www.mrtg.org/`.

[29] "NOCOL- network monitoring software", `http://www.netplex-tech.com/software/nocol/`.

[30] M. Dilman and D. Raz, "Efficient reactive monitoring", *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 668–676, 2002.

[31] M. Su, K. Thulasiraman, and A. Das, "A scalable on-line multilevel distributed network fault detection/monitoring system based on the SNMP protocol", in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, 2002, pp. 1971–1975.

[32] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony, "Hierarchical network management: A scalable and dynamic mobile agent-based approach", *Computer Networks*, vol. 38, no. 6, pp. 693–711, 2002.

[33] A. Liotta, G. Pavlou, and G. Knight, "Exploiting agent mobility for large scale network monitoring", *IEEE Network*, vol. 16, no. 3, pp. 7–15, 2002.

[34] A. Tripathi, T. Ahmed, S. Pathak, M. Carney, and P. Dokas, "Paradigms for mobile agent-based active monitoring of network systems", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2002, pp. 65–78.

[35] K. Boudaoud, H. Labiod, R. Boutaba, and Z. Guessoum, "Network security management with intelligent agents", in *Proceedings of IEEE/IFIP*

Network Operations and Management Symposium (NOMS), 2000, pp. 579–592.

[36] M. Zapf, K. Herrmann, and K. Geihs, "Decentralized SNMP management with mobile agents", in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 1999, pp. 623–635.

[37] D. Breitgand, D. Dolev, D. Raz, and G. Shaviner, "Facilitating efficient and reliable monitoring through HAMSA", in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2003, pp. 263–276.

[38] E. Al-Shaer, "A dynamic group management framework for large-scale distributed event monitoring", in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2001, pp. 361–374.

[39] Y. Chen, "Study on the prevention of SYN flooding by using traffic policing", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2000, pp. 593–604.

[40] M. Kim, H. Kang, S. Hong, S. Chung, and J. Hong, "A flow-based method for abnormal network traffic detection", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2004, pp. 599–612.

[41] A. Pasupulati, J. Coit, K. Levitt, S. Wu, S. Li, J. Kuo, and K. Fan, "Buttercup: On network-based detection of polymorphic buffer overflow vulnerabilities", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2004, pp. 235–248.

[42] M. Polychronakis, E. Markatos, A. Øslebø, and K. Anagnostakis, "Design of an application programming interface for IP network monitoring", in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2004, pp. 483–496.

[43] "Snort.org", `http://www.snort.org/`.

[44] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attack", in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2002, pp. 1530–1539.

[45] C. Bettstetter and C. Wagner, "The spatial node distribution of the random waypoint mobility model", in *Proceedings of Workshop on Mobile Ad hoc Networks (WMAN)*, 2002, pp. 41–58.

[46] G. Resta and P. Santi, "An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks", in *Proceedings of ACM International Workshop on Principles of Mobile Computing (POMC)*, 2002, pp. 44–50.

[47] A. Jardosh, E. Belding-Royer, K. Almeroth, and S. Suri, "Towards realistic mobility models for mobile ad hoc networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2003, pp. 217–229.

[48] M. Hollick, T. Krop, J. Schmitt, H.-P. Huth, and R. Steinmetz, "Modeling mobility and workload for wireless metropolitan area networks", *Computer Communications*, vol. 27, no. 8, pp. 751–761, 2004.

[49] W. Hsu, K. Merchant, H. Shu, C. Hsu, and A. Helmy, "Weighted waypoint mobility model and its impact on ad hoc networks", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 1, pp. 59–63, 2005.

[50] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2002, pp. 107–118.

[51] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004, pp. 187–201.

[52] D. Tang and M. Baker, "Analysis of a metropolitan-area wireless network", *Wireless Networks*, vol. 8, no. 2/3, pp. 107–120, 2002.

[53] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan, "Characterizing user behavior and network performance in a public wireless LAN", *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 195–205, 2002.

[54] A. Nasipuri, neda R. Casta and S. Das, "Performance of multipath routing for on-demand protocols in mobile ad hoc networks", *Mobile Networks and Applications*, vol. 6, no. 4, pp. 339–349, 2001.

[55] W. Sun, H. Yamaguchi, K. Yukimasa, and S. Kusumoto, "GVGrid: A QoS routing protocol for vehicular ad hoc networks", in *Proceedings of IEEE International Workshop on Quality of Service*, 2006, pp. 130–139.

[56] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, "MDDV: A mobility-centric data dissemination algorithm for vehicular networks", in *Proceedings of ACM Workshop on Vehicular Ad Hoc Networks (VANET)*, 2004, pp. 47–56.

[57] C. Toh, "Associativity-based routing for ad-hoc mobile networks", *Journal on Wireless Personal Communications*, vol. 4, no. 2, pp. 103–139, 1997.

[58] R. Dube, C. Rais, K. Wang, and S. Tripathi, "Signal stability based adaptive routing (SSA) for ad hoc mobile networks", *IEEE Personal Communications Magazine*, vol. 4, no. 1, pp. 36–45, 1997.

[59] R. Sisodia, B. Manoj, and C. Murthy, "A preferred link based routing protocol for ad hoc wireless networks", *Journal of Communications and Networks*, vol. 4, no. 1, pp. 14–21, 2002.

[60] D. Johnson, "Routing in ad hoc networks of mobile hosts", in *Proceedings of Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1994, pp. 158–163.

[61] T. Imielinski and J. Navas, "GPS-based geographic addressing, routing, and resource discovery", *Communications of the ACM*, vol. 42, no. 4, pp. 86–92, 1999.

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

[63] I. Stojmenovic, "Position based routing in ad hoc wireless networks", *IEEE Communications Magazine*, vol. 40, no. 7, pp. 128–134, 2002.

[64] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (DREAM)", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1998, pp. 76–84.

[65] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000, pp. 243–254.

[66] I. Stojmenovic, A. Ruhil, and D. Lobiyal, "Voronoi diagram and convex hull based geocasting and routing in wireless networks", in *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, 2003, pp. 51–56.

[67] X. Lin and I. Stojmenovic, "Location-based localized alternate, disjoint and multi-path routing algorithms for wireless networks", *Journal of Parallel and Distributed Computing*, vol. 63, no. 1, pp. 22–32, 2003.

[68] S. Basagni, I. Chlamtac, and V. Syrotiuk, "Location aware, dependable multicast for mobile ad hoc networks", *Computer Networks*, vol. 36, no. 5–6, pp. 659–670, 2002.

[69] Y. Ko and N. Vaidya, "Geocasting in mobile ad hoc networks; location-based multicast algorithms", in *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999, pp. 101–110.

[70] Y. Ko and N. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1998, pp. 66–75.

[71] W. Liao, Y. Tseng, K. Lo, and J. Sheu, "Geogrid: A geocasting protocol for mobile ad hoc networks based on grid", *Journal of Internet Technology*, vol. 1, no. 2, pp. 23–32, 2000.

[72] C. Chang, C. Chang, and S. Tu, "Obstacle-free geocasting protocols for single/multi-destination short message services in ad hoc networks", *Wireless Networks*, vol. 9, no. 2, pp. 143–155, 2003.

[73] B. An and S. Papavassiliou, "Geomulticast: Architectures and protocols for mobile ad hoc wireless networks", *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 182–195, 2003.

[74] T. Camp and Y. Liu, "An adaptive mesh-based protocol for geocast routing", *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 196–213, 2003.

[75] J. Boleng, T. Camp, and V. Tolety, "Mesh-based geocast routing protocols in an ad hoc network", in *Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2001, pp. 184–193.

[76] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanner for routing in mobile networks", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001, pp. 45–55.

[77] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003, pp. 267–278.

[78] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system", in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2000, pp. 775–784.

[79] N. Patwari and A. Hero, "Using proximity and quantized RSS for sensor localization in wireless networks", in *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003, pp. 20–29.

[80] D. Niculescu and B. Nath, "VOR base stations for indoor 802.11 positioning", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004, pp. 58–69.

[81] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000, pp. 32–43.

[82] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking moving devices with the cricket location system", in *Proceedings of the ACM International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2004, pp. 190–202.

[83] A. Savvides, C. Han, and M. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 166–179.

[84] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices", *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, 2000.

[85] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks", in *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2003, pp. 81–95.

[86] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003, pp. 201–212.

[87] H. Lim and J. Hou, "Localization for anisotropic sensor networks", in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2005, pp. 138–149.

[88] C. Wang and L. Xiao, "Locating sensors in concave areas", in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–12.

[89] T. Umedu, K. Yasumoto, A. Nakata, H. Yamaguchi, and T. Higashino, "Middleware for synchronous group communication in wireless ad hoc networks", in *Proceedings of Communications and Computer Networks (CCN)*, 2002, pp. 48–53.

[90] T. Umedu, Y. Terashima, K. Yasumoto, A. Nakata, T. Higashino, and K. Taniguchi, "A language for describing wireless mobile applica-

tions with dynamic establishment of multi-way synchronization channels", in *Proceedings of International Conference on Formal Methods Europe (LNCS2391)*, 2002, pp. 607–624.

[91] A. Tanenbaum and M. Steen, *Distributed systems: Principles and paradigms*, Prentice Hall, 2002.

[92] "Tiers topology generator", `http://www.isi.edu/nsnam/ns/ns-topogen.html/#tiers`.

[93] "SINET", `http://www.sinet.ad.jp/topology`.

[94] S. Shah and K. Nahrstedt, "Predictive location-based QoS routing in mobile ad hoc networks", Tech. Rep., University of Illinois at Urbana-Champaign, 2001.

[95] William Su, Sung-Ju Lee, and Mario Gerla, "Mobility prediction and routing in ad hoc wireless networks", *International Journal of Network Management*, vol. 11, no. 1, pp. 1099–1190, 2001.

[96] G. Riley, "The Georgia Tech network simulator", in *Proceedings of ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, 2003, pp. 5 – 12.

[97] A. McDonald and T. Znati, "Predicting node proximity in ad-hoc networks: A least overhead adaptive model for selecting stable routes", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000, pp. 29–33.

[98] M. Saito, J. Tsukamoto, T. Umedu, and T. Higashino, "Evaluation of inter-vehicle ad hoc communication protocol", in *Proceedings of IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2005, pp. 78–83.

[99] E. Teramoto, M. Baba, H. Mori, H. Kitaoka, I. Tanahashi, Y. Nishimura, T. Saito, Y. Takizawa, T. Sawa, and H. Ooe, "Prediction of conditions for the nagano olympic winter games using traffic simulator NETSTREAM", in *Proceedings of World Congress on Intelligent Transport Systems*, 1998, pp. 1801–1806.

[100] T. Tracy, B. Jeff, and D. Vanessa, "A survey of mobility models for ad hoc network research", *Wireless Communication & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends, and Applications*, vol. 2, no. 5, pp. 483 – 502, 2002.

[101] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research", *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[102] R. Flury and R. Wattenhofer, "MLS: An efficient location service for mobile ad hoc networks", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006, pp. 226–237.

[103] M. McGuire, "Stationary distributions of random walk mobility models for wireless ad hoc networks", in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005, pp. 90–98.

[104] D. Son, A. Helmy, and B. Krishnamachari, "The effect of mobility-induced location errors on geographic routing in mobile ad hoc and sensor networks: Analysis and improvement using mobility prediction", *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 233–245, 2004.