IEICE TRANS. COMMUN., VOL.E90–B, NO.11 NOVEMBER 2007

LETTER *Special Section on Next Generation Network Management*

Development of a Topology Controllable Testing Environment for Mobile Ad Hoc Network Software*

Atsushi KAWANO[†], ***Student Member***, **Tomoyuki OHTA**[†a)], **Kaori MAEDA**[†], **Kenji ISHIDA**[†], ***and*** **Yoshiaki KAKUDA**[†], ***Members***

SUMMARY A mobile ad hoc network is an autonomous wireless network which consists of mobile nodes without any base stations. Many routing schemes and services have been proposed for mobile ad hoc networks. However, since these schemes tend to be evaluated through simulation experiments, it is not known whether they work effectively in real environments or not. Therefore, in order to verify their practical use in mobile ad hoc networks, it is necessary to perform field experiments using actual mobile nodes. If the network size is large, it is difficult to perform field experiments due to problems with limited battery, difficulty of topology control and so on. Realization of rapid topology change of the ad hoc networks topology is especially difficult. In order to solve this problem, this paper proposes a testing environment for mobile ad hoc network software, which emulates field experiments in wired networks.

key words: *mobile ad hoc networks, software testing*

1. Introduction

Recently, many routing algorithms and applications have been proposed for mobile ad hoc networks. However, most of them are only evaluated through simulation experiments and are not verified using actual mobile nodes in real environments, which are dynamic and resource-constrained. It is difficult to flexibly realize the ad hoc network in such real environments for testing its software. Realization of rapid topology change of the ad hoc network topology is especially difficult. Therefore, many network emulators for mobile ad hoc networks have been proposed [2]–[4]. In these emulators, the node mobility model is not considered as well. Since nodes always move in ad hoc networks, the node mobility model is the most important factor to test the ad hoc network software and affects the test results.

The above emulators are *scenario-driven*. The scenario, in which the location and migration information of nodes is predetermined, is obtained from the random waypoint model. However, in fact, as known in the car-to-car mobile ad hoc network applications [5], each node representing a car controls the current moving speed and direction by events such as making speed up or down and turning to the right or left according to the situation of its neighboring nodes. As a result, the network topology is autonomously changed due to such events even during execution of the emulation.

In order to solve the above problem, we have proposed a preliminary testing environment for mobile ad hoc network software [1]. This paper proposes a topology controllable testing environment for mobile ad hoc network software which adopts *scenario-independent* mechanism. In this mechanism, the network topology can be changed by events which are not predetermined.

2. Related Works

This section shows some of the most important related network emulators for mobile ad hoc networks. NEMAN [2] consists of a "Topology Manager" and multiple application process. It can emulate the wireless network which consists of hundreds of nodes. Topology Manager manages the topology and the packet delivery. JEmu [3] consists of an emulation engine and multiple nodes. All packets from each node are delivered to the emulation engine. The emulation engine decides whether the packet are delivered or discarded. The network topology is intensively managed by the emulation engine. MobiEmu [4] consists of several tested slave nodes and one master node. The master node and the slave nodes have the identical scenario file. The master node delivers a message to the slave nodes to inform the change of the topology information which the master server manages. The slave node that receives the message performs the packet filter based on the topology information. Thus, the network topology is emulated.

NEMAN and JEmu operate many nodes with one device. It is easy to experiment using a lot of nodes, in which the restriction of the resource is the different from real environments. MobiEmu operate one node with one device. The restriction of the resource is the same as real environments. Thus, these emulators can almost realize a real environment. However, these emulators are *scenario-driven*. Thus, the scenario of node movement are predefined according to the node mobility model and not changed during execution of the emulation. Unlike the previous emulators, this paper proposes a topology controllable testing environment for mobile ad hoc network software which adopts *scenario-independent* mechanism, which enables change of the net-

Manuscript received March 19, 2007.
Manuscript revised June 13, 2007.

[†]The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731-3194 Japan.

*This work was supported in part by the Ministry of Internal Affairs and Communications of Japan under Grant-in-Aid for SCOPE-C (No.052308002) and the Ministry of Education, Science, Sports and Culture of Japan under Grant-in-Aid for Scientific Research (C) (No.18500062) and Young Scientists (B) (No.18700070) under their research grants.

a) E-mail: ohta@ce.hiroshima-cu.ac.jp

DOI: 10.1093/ietcom/e90–b.11.3104

work topology caused by control of nodes even during execution of emulation.

3. Topology Controllable Testing Environment

This paper proposes a topology controllable testing environment for mobile ad hoc network software. In the proposed testing environment, the topology of a mobile ad hoc network can be virtually configured in a wired network. It consists of one positioning server and multiple testing nodes as shown in Fig. 1. The target application software for testing is assumed to be run on testing nodes. The positioning server manages virtual information of the location and migration (moving speed and destination) of each node and distributes it to testing nodes.

In order to virtually configure the topology of a mobile ad hoc network in the wired network, each testing node determines the neighboring nodes based on the location information which is sent from the positioning server to each testing node. In addition, each testing node can autonomously change its location information by sending the location information from the testing node to the positioning server. Therefore, the proposed testing environment adopts scenario-independent mechanism, which is *event*-driven in the sense that the testing nodes can autonomously alter the location and migration information which includes the maximum moving speed and the destination by its events.

Each testing node can request the positioning server to change the location and migration information. If the positioning server receives the request from the testing node, it updates the location and migration information according to the request. This mechanism can therefore realize the emulation of applications such as car-to-car mobile ad hoc networks using wired networks. This is the original and novel issue of the proposed testing environment.

3.1 Positioning Server

The positioning server manages the location and migration information as follows. At first, it virtually configures a field (called a virtual field) and nodes (called virtual nodes) for a mobile ad hoc network. Next, virtual nodes are set in the virtual field by assigning the location information to virtual nodes. Then, moving speed of each node is set randomly as migration information within the given range of the moving speed. After each virtual node migrates to the specified destination, the same procedures are repeated until the emulation stops.

In the proposed testing environment, one testing node is used for one virtual node. As shown in Fig. 2, the positioning server periodically distributes the location information to testing nodes. As a result, each testing node can get the location information in the virtual field which is uniquely determined by the positioning server. Control messages which are exchanged between the positioning server and testing nodes are as follows.

Join message: When a node joins the testing environment, the node sends it to the positioning server. The positioning server which received it creates a virtual node which is assigned to the new testing node in the virtual field.

Leave message: When a testing node leaves from the testing environment, it sends this message to the positioning server. The positioning server which received it deletes the virtual node which is assigned to the testing node in the virtual field.

Configuration message: The location and migration information for virtual nodes are contained in this message. When the positioning server which received the Join message permits a new node to join, the positioning server sends it back to the new node. The new node which received it becomes a new testing node in the testing environment.

Position message: The location and migration information for virtual nodes are contained in this message. It is periodically sent from the positioning server to testing nodes. Each testing node gets its own virtual location information from the positioning server. When a testing node changes the current location and migration information of its own or the other testing node, the testing node sends it to the positioning server.

Map message: When a testing node changes its own migration information, the testing node sends this message to the positioning server. The positioning server which received it distributes the Position message which includes the migration information to all testing nodes.

When a node joins the testing environment, it sends the Join message to the positioning server. The positioning server which received the Join message sends the Configuration message back to the node. The node which received the Configuration message becomes the testing node in the testing environment. The positioning server assigns a virtual node to the testing node and periodically sends Position

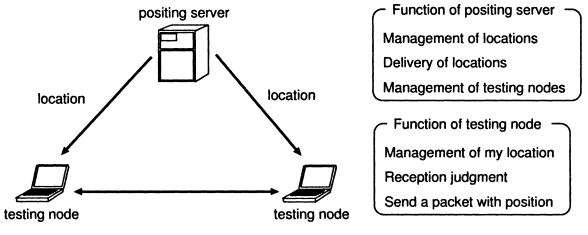

Fig. 1 Each function in topology controllable testing environment.

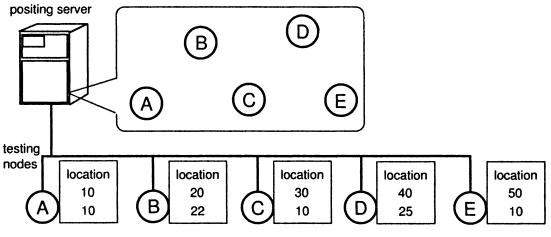

Fig. 2 Example of the communication between the positioning server and testing nodes.

message to inform the location information to the testing node. When a testing node leaves from the testing environment, it sends the Leave message to the positioning server.

3.2 Communication Mechanism between Testing Nodes

Communication mechanism between testing nodes is as follows. Packets for mobile ad hoc network software only are exchanged among testing nodes. Each testing node has the transmission range and the location information by receiving Configuration and Position messages from the positioning server. A source testing node sends a packet with the current location information to the other testing nodes. Each testing node which received the packet gets the location information of the source testing and compares it with the current location information. If the location of the source node is within the transmission range, each testing node emulates reception of the packet in the mobile ad hoc network. On the other hand, if the location of the source is not within the transmission range, each testing node discards the packet.

4. Implementation

4.1 Positioning Server

Figure 3 shows a graphical user interface of the positioning server. As shown in Fig. 3, developers can select the execution mode, set parameters for testing environment and check the current network topology through this interfaces which are indicated by (a), (b), and (c), respectively. In order to test the ad hoc network software, there are the following three execution modes in the testing environment.

Custom mode: Developers can test the ad hoc network software under condition that the initial placement of virtual nodes in the virtual field is given and the network topology is static. Developers create the setting file of the initial placement of virtual nodes and make a positioning server read it to perform the test.

Custom-Random mode: Developers can test the ad hoc network software under condition that the initial placement of virtual nodes in the virtual field is given. A network topology is dynamically changed as time proceeds since virtual nodes are migrated by the positioning server. Developers utilize a setting file for the initial placement of virtual nodes like Custom mode and set the maximum moving speed of nodes as the node migration information through the GUI (Fig. 3(b)).

Random mode: Developers can test the ad hoc network software under condition that the initial placement of virtual nodes in the virtual field is randomly placed and the network topology is randomly changed by the positioning server. In addition to the maximum moving speed of nodes like Custom-Random mode, developers set the number of nodes which can join the testing environment and the field size of the virtual field (Fig. 3(b)).

In each mode, developers can change and control the location and migration information of each testing node through the interface (Fig. 3(d)).

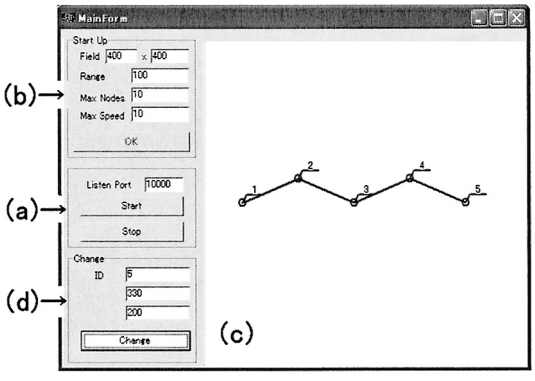

Fig. 3 GUI of a positioning server.

4.2 Testing Node

In the proposed testing environment, the function of each testing node is implemented in AdHocDevice layer of AdHocEngine which we have developed [6]. AdHocEngine provides multihop wireless communication for applications in mobile ad hoc networks. Owing to AdHocEngine, the source node can communicate with the destination node via the other nodes. Figure 4 shows the architecture including AdHocEngine framework for mobile ad hoc networks. Each layer of AdHocEngine is defined as follows.

AdHocCtrl manages AdHocTransport and AdHocDevice and provides datagram communication and packet forwarding function. It forwards packets to the next node based on the information from AdHocRouting. AdHocTransport selects a transport protocol for mobile ad hoc networks and provides reliable communication according to the request of the application. AdHocRouting provides routing information to AdHocCtrl and is defined as the abstract class. A developer can implement a routing protocol by inheriting this class. AdHocDevice provides an interface for a wireless device such as IEEE802.11 to AdHocCtrl and implements the function of sending and receiving packets to/from neighboring nodes.

In each testing node, when the AdHocDevice receives the information to change the location and the migration information from the application layer, it sends the MAP message to the positioning server to request the change of the

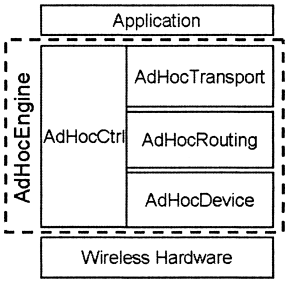

Fig. 4 Structure of AdHocEngine.

location and the migration information. As a result, developers can change testing nodes' location and moving speed not only from GUI interface but also from the application on the testing node.

5. Case Study

This section describes the case study to develop the chat application for the ad hoc network environments using the proposed testing environment. In this application, each user can talk with the other users through the ad hoc network. In case that the developer tests the application in real environments, the developer deploys multiple mobile computers which have IEEE802.11b in the field, and then tests whether they are able to communicate between any two nodes or among more than two nodes. In addition, whenever the developer controls the topology of the ad hoc network to test the application in the various environments, all or some mobile nodes must be moved by hand. Therefore, it costs much manpower and money to test the application in the real environments.

On the other hand, if the developer utilizes the proposed testing environment to test the application for the ad hoc environments, the developer can easily control the topology of a mobile ad hoc network. Since the proposed testing environment provides the topology controllable mechanism in the virtual field, the developer can easily control only the position of a node to create the topology which the developer intends. Therefore, the developer can easily test the application for the ad hoc network environments without moving multiple mobile nodes.

A prototype of the proposed testing environment consists of one laptop computer for the positioning server and five laptop computers for testing nodes (that is, six laptop computers are totally used). The GUI (Fig. 3(c)) represents the virtual field which the positioning server manages. Here, node 1 communicates with node 5 in the ad hoc network topology as shown in Fig. 3. Figure 5 shows the windows of the chat application working on nodes 1 and 5, respectively. In addition, the developer can change the position of each node in the virtual field through the GUI (Fig. 3(d)). Therefore, the developer can easily control the ad hoc network topology as shown in Fig. 3 and test the application in various network topology using the proposed testing environments.

The application developed by the proposed testing environment can work in the real environments by exchanging the function of AdHocDevice.

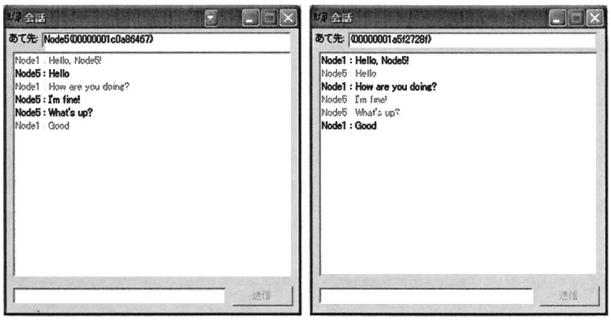

Fig. 5 Chat windows for the application working on node 1 (left window) and node 5 (right window), respectively.

6. Conclusion

This paper has proposed a topology controllable testing environment for mobile ad hoc network software which adopts scenario-independent mechanism to reconfigure the ad hoc network topology in wired networks. It realizes a flexible mobility model, in which each node can autonomously change its location and migration information. Therefore, it can be applied to wide applications in comparison with existing emulators. Consequently, it is expected that the mobile ad hoc network software can be effectively developed by utilizing the proposed testing environment.

References

[1] A. Kawano, D. Oka, Y. Kubo, S. Yamashita, K. Maeda, T. Ohta, K. Ishida, and Y. Kakuda, "A topology controllable testing environment for mobile ad hoc network software," Proc. 2nd Int'l Conf. on Mobile Ad-hoc and Sensor Networks (MSN 2006), Lecture Notes in Computer Sciences (LNCS), vol.4325, pp.808–819, Springer, 2006.

[2] M. Puzar and T. Plagemann, "Neman: A network emulator for mobile ad-hoc networks," Proc. 8th Int'l Conf. on Telecommunications, vol.1, pp.155–161, 2005.

[3] J. Flynn, H. Tewari, and D. O'Mahony, "Jemu: A real-time emulation system for mobile ad-hoc networks," Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference, 2002.

[4] Y. Zhang and W. Li, "An integrated environment for testing mobile ad-hoc networks," Proc. 3rd ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC2002), pp.104–111, 2002.

[5] W. Chen and S. Cai, "Ad hoc peer-to-peer network architecture for vehicle safety communications," IEEE Commun. Mag., vol.43, no.4, pp.100–107, 2005.

[6] S. Yamashita, R. Furukawa, T. Ohta, H. Kojima, and Y. Kakuda, "Development of testbed framework for ad hoc networks," Proc. Int'l Symp. on Wireless Personal Multimedia Communications (WPMC2005), vol.1, pp.383–387, 2005.