A New Event-Driven Network Simulator for Delay-Tolerant Networks (DTNs)

Euiyul Ko
Dept. of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
key@cnlab.kaist.ac.kr

Hanjin Park
Dept. of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
hjpark@cnlab.kaist.ac.kr

Ikjun Yeom
Information & Communication Eng.
Sungkyunkwan University
Suwon, South Korea
ikjun@skku.edu

ABSTRACT

As wireless networking technologies have evolved, wireless networks are deployed in wider areas. In some of them, end-to-end paths may not be guaranteed due to high mobility or low density of network nodes. Delay/Disruption Tolerant Networks (DTNs) or Intermittently Connected Networks (ICNs) are proposed to handle packet delivery in those networks. In DTNs, network simulation may play a critical role for developing new technologies since it is very expensive and time-consuming to perform real experiments. In this paper, we present a new simulation tool for DTNs. This tool is developed with an event-driven manner to accommodate discrete events in packet delivery processes in DTNs, and also it is highly flexible and easily extensible for developing new protocols. We validate the tool with well-known mathematical models of DTNs.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network simulations

General Terms

Network simulation

Keywords

Delay/Disruption Tolerant Network, Event-driven simulator, Simulator validation

1. INTRODUCTION

With the increasing use of wireless mobile devices, Mobile Ad-hoc Networks (MANETs) have been studied actively in a last decade. Unlike traditional infrastructure-based TCP/IP network environments, a MANET is a network formed with a set of mobile nodes without pre-configured infrastructure. In a MANET, each mobile node participates to organize a network as a router or an end user by relaying/forwarding packets. The main assumption of traditional networks is that there always exists a fully connected path among nodes to communicate between a source and a destination. In other words, there should exist at least one contemporary fully connected path between end users to delivery packets. Some subsets of a MANET, however, cannot satisfy this assumption, for example, a network with high mobility of mobile nodes, low density of mobile nodes, frequently partitioning, or high bit error rate. In order to overcome this characteristic, a new type of network called as Delay/Disruption Tolerant Network (DTN) or Intermittently Connected Network (ICN) [8] has been suggested.

Real experiments, for capturing the state or behavior of DTN, are very difficult because of difficulties of implementation in physical test-bed and construction of experiment environments. Experiments, moreover, can be affected by other unconsidered factors, which lead to wrong conclusion. Because simulations can abstract behaviors of DTN protocol and simplify characteristics of them, simulations are useful tools to analyze behaviors of DTN protocols and to help to concentrate specific measures that are interested in originally.

There are two time advance approaches in simulation: a) time-driven simulator; and b) event-driven simulator. The former, a time-driven approach, advances time with a fixed interval Δt for continuous systems. After Δt, the simulator updates the state variables. It is the most widely known approach in simulations of natural world. *ONE*[1] (Opportunistic Networking Environment) simulator [11] is a well known DTN simulator based on a time-driven approach. The latter, a event-driven approach, advances time with the next-event time. Polling the next event, a simulator processes that event and it updates time to next event time. This approach is for the case of discrete systems. The *DTNSim2*[2] is a example based on an event-driven approach.

The event-driven approach has advantages compare with the time-driven approach. The result of time-driven approach is very sensitive to a fixed interval Δt. For capturing all events, Δt is the minimum time between two continuous events. If Δt is larger than the minimum time, the simulator does not capture all events in $[t, t+\Delta t]$, which can lead wrong results. On the contrary, if Δt is smaller than the minimum time, the simulator capture every event in the discrete system, but simulation performance, such as simulation time,

[1] http://www.netlab.tkk.fi/tutkimus/dtn/theone/
[2] http://watwire.uwaterloo.ca/DTN/sim/

SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650

resource usage, and so on, is down. In time-driven approach, two events between Δt are handled as events which start the same time although they have different starting time. The event-driven approach also has the advantage that periods of inactivity can be skipped over by leaping the time from the current event time to the next event time, which can raise up simulation performance. Since the *ONE* simulator is based on time-driven approach, it has mentioned weaknesses. The *DTNSim2* has advantages because it is based on event-driven approach. Since various mobility models, however, was not implemented in it, it cannot test them.

In this paper, we describe our new simulator based on event-driven approach. It is written by JAVA language. We first make frameworks for easy extension of other protocols. For processing events, first, event generation modules are required. For simulation, we have defined three different events, such as contact event, message creation event, and transfer event. Generated events are processed in the scheduler. We have defined the scheduler which processes events in order of time and manages event queue. Finally, we build a mobile node which is constructed a node class (application layer), a transport class (transport layer), and a routing class (network layer). We did not develop MAC and PHY layer since they does not critically affect DTN performance and MAC and PHY layer may have an unpredictable effect on results. These classes are based on frameworks, so new protocols in transport or network layer can be easily implemented by our or other researchers.

The rest of paper is organized as follow: In Section 2, we summarize related works on DTN and simulations. We describe our simulator detailed in Section 3. In Section 4, we validate our simulator's results compared with analytical models, and describe our performance. In Section 5, more works to do are described. Finally, we conclude this paper with summary in Section 6.

2. RELATED WORKS

DTN has been actively researched for last several years. There is a well-known research group for studying DTN protocols that is *Delay Tolerant Networking Research Group (DTNRG) [2]*. In [2], a lot of researchers have proposed several protocols for DTN. To evaluate performance of these protocols, the real experiments are often difficult because of difficulties of the implementation, cost for constructing test-bed, and biased results by limited samples. Therefore, simulations have been used for evaluating performance.

There are several simulation tools have been developed so far to help to understand the underlying performance of routing protocols in DTN, such as *ONE* [11], *DTNSim2* [10], and so on. *ONE* written by JAVA language is easy to simulate some scenarios, and provide many supporting mobility models and routings with optional GUI interfaces. Moreover, *ONE* system is updated frequently to improve its quality and include new routing protocols for DTN.

ONE, however, have been developed by the time-driven approach. Even though it is called itself as a discrete event simulator and generates several events (e.g. message creation events, message transferred events, and so on), there is a global clock, which is advanced time with a system configured fixed interval Δt, in it to capture the events. According to the size of fixed interval Δt, the simulator may generate more precise results with very long computation times or inaccurate results with reasonable computation times. There is a trade-off relationship between efficiency and precision in the time-driven simulator as it. General events are occurred in DTN simulations, such as message creation events, contact events between nodes, and message transfer events, are not continuous time event but mainly discrete events. In these discrete systems, the event-driven simulation approach is more appropriate than the time-driven simulation.

DTNSim2 is a simulator written in JAVA language and is developed several routings appeared in [10] paper, such as First Contact (FC), Minimum Expected Delay (MED), Earliest Delivery (ED), Earliest Delivery with Local Queue (EDLQ), Earliest Delivery with All Queue (EDAQ), and Linear Program (LP). It can be configured with simple scripts with contact schedules and traffic parameters, is easy to execute simulation, and was developed by the event-driven approach. It, however, was developed at 2006 and after that time there is no updates/evolutions to improve its quality or to fix bugs. Moreover, it lacks supporting various mobility models and is only working properly in some limited scenario examples.

There are also several simple simulators embedded in ns-2 [1] simulator, which is the most popular network simulating tool, to implement DTN routing, such as epidemic routing simulator [14] and nsdtn-1 [3]. In [14], they implemented only epidemic routing in ns-2 simulator. However, the epidemic simulator was developed in very old ns-2 version, 2.17b, and is working only limited scenarios. With MAC and PHY modules in ns-2, the result of simulation can be biased or affected by non-routing protocols. The epidemic routing simulator, moreover, [14] is not working properly in some scenario. In [3], the nsdtn-1 simulator was developed with the fixed mobility model that is not appropriated in general DTN environments which are mainly focused with mixed end nodes between mobile users and fixed infrastructures.

As discussed above, the existing simulators may give inaccurate results in the time-driven approach case or may not handle many mobility models or may lack extensibility to include more features. Our simulator has the following strong points compared previous simulators: a) Provide frameworks for easy extension with routing and mobility. Each layer has been developed based on framework as a class so that any researcher can easily implement the protocol to be used in the layer or exchange it, such as application layer, transport layer, network layer; b) With event driven approach, our simulator can generate the precise result with reasonable computation time; and c) With excluding MAC and PHY part, we can show the abstraction behavior of routing and above layer protocols in DTN.

3. A NEW EVENT-DRIVEN SIMULATOR FOR DTNS

The event-driven simulator progresses time with the next event time. For these, all events should be generated before simulation is started. The number of event type is three, *Contact event, Message Creation event*, and *Transfer event*. Contact event is generated when a node encounters another node. Message creation event is produced by traffic generation. Transfer event is generated when practical transmission is occurred.

First, we have developed event generation functions, and we have made frameworks for a simulator and new imple-

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650

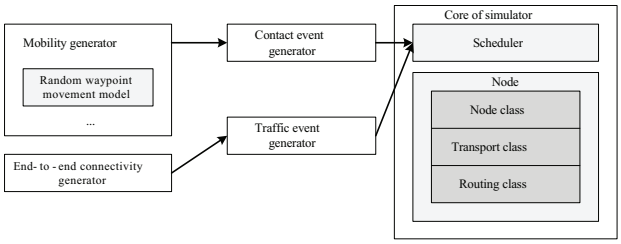


Figure 1: Overview of the simulator

mented protocols. We have also developed basic features of DTN and protocols in our simulator. The simulator first generates all events, such as contact event, message creation event, and transfer event, to be processed during a simulation. The simulation scheduler processes events in order of event execution time. The simulation states are updated at every event executions.

The simulator consists of five main parts: a) mobility generator; b) end-to-end connectivity generator; c) contact event generator; d) traffic event generator; and e) core of simulator, which includes the event scheduler and the node agent. Mobility generator and end-to-end connectivity generator are independent of other parts. These parts and their relations are shown in Figure 1.

Nodes' movement is generated by mobility generator. Currently, we implement a random waypoint movement model. An end-to-end connectivity generator makes connections between two nodes which are used in traffic generator. Contact event generator creates contact events using results of mobility generator. Message creation events are made by traffic event generator. Currently, we implement CBR traffic with fixed end-to-end connectivity, and random end-to-end connectivity. Contact and message creation events are scheduled in the scheduler of core of simulator. A simulation is performed with configuration files. In a configuration file, we can choose a traffic pattern, scenario, and other settings.

In the rest of this section, we describe each part more detailed and how to add new protocols or mobility models.

3.1 Mobility Generator

Mobility is important to simulate the protocol and evaluate its protocol performance in DTNs. Mobility models characterize movement patterns of nodes. Currently, there are some kinds of mobility models, for example, Random Walk (RW) model, Random Waypoint (RWP) model, Random Direction (RD) model, Brownian model, Random Gauss-Markov model, and so on.

Our simulator includes nodes' movement generator using a mobility model. Currently, we have implemented RWP model, which is popular mobility model while it have some known defects. The result of generator is used a contact event generator.

To add a new mobility model, you can easily implement it as long as the result of implemented model is matched to pre-implemented format. Our result format is formed as Figure 2. A format is organized as follow: a) start time when current move is started; b) end time when current move is ended; c) the node index; d) speed on X axis in the simulation world; e) an initial position on X axis; f) speed on Y axis in the simulation world; and g) an initial position on Y axis. This format is only applied if a path is the straight

Start time	End time	# of node	Speed of X axis	Start position of X axis	Speed of Y axis	Start position of Y axis

Figure 2: Format of a result of mobility generator

line, but it is enough to process other mobility models.

3.2 Contact Event Generator

The contact event generator makes contact events based on a result of mobility generator. Contact events consists of two nodes, which meet each other, meeting start time, and duration of meeting.

Finding a contact point in two nodes during $[t_1, t_2]$ is simple. Let two nodes' path functions be $f_a(t)$ and $f_b(t)$ in $[t_1, t_2]$. At time t in $[t_1, t_2]$, a distance $d(t)$ between two nodes is $\sqrt{(f_a(t) - f_b(t))^2}$. Checking whether $d(t)$ is smaller or bigger than a transmission range, we can determine whether a contact event in two nodes are occurred during $[t_1, t_2]$.

In the contact event generator, first, it makes all interval through a mobility scenario, and it finds contacts between nodes. If a contact is found, it generates a contact event and pushes it into a scheduler.

3.2.1 Transfer Event

When two nodes meet each other, they try to transmissions each other. If any node in two is already connected another node except two, first two nodes do not transfer their messages. Solving this problem, transfer events are generated when a contact event is processed.

If two nodes do not transfer messages to other nodes, the first transfer event is generated and pushed in a scheduler. When this event is processed, a next transfer event between these two nodes is generated until there is no transmitted messages in two nodes.

Otherwise, which any node of two has another meeting node, a transfer event is not generated until previous transfer events is end. In this case, the contact event is postponed until previous transfer events is end.

3.3 Traffic Event Generator

Messages are generated by the traffic event generator. We have implement two message generators: a) CBR traffic with fixed senders and receivers, and b) CBR traffic with random senders and receivers. A CBR traffic with fixed sources and destinations uses a result of the end-to-end connectivity generator. The reason why we use the end-to-end connectivity generator is that the simulator must produce the same results if simulations do with the same environments. In a CBR traffic with random senders and receivers, a sender and a receiver of a message is determined when a message is generated. All message generation events generated by the traffic event generator are made at the start time of simulation.

Adding new traffic patterns is not difficult. For adding new traffic patterns, the new generator has to make intervals and generate a message in each interval. Interval calculations depend on new traffic patterns, and you can easily implement them.

3.4 The Core of Simulator

The core of simulator is consisted of a scheduler and a node. A scheduler manages events, such as contact, transfer,

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650

and message creation events. A node includes a node class (application layer), a transport class (transport layer), and a routing class (network layer).

A scheduler consists two queues, which one queue is for contact and transfer events, and another queue is for message creation, enqueue methods, and a scheduling method. If all events are managed in one queue, simulation performance is down at event generation time because events are not made in sequence of time. Enqueue methods insert a event to an queue at right position. A scheduling method picks out a event in queues and processes it.

A node is the basic agent in the simulator. At processing a simulation, a set of nodes is created and manages all data used by a simulation. A node consists of a node class (application layer), a transport class (transport layer), and a routing class (network layer). We do not implement MAC and PHY layers. While these two layers are important in legacy network schemes, such as wired, wireless, and ad-hoc networks, they are less influential in DTNs since competitions among nodes are not often compared with legacy networks.

A node class represents an application layer program. In this class, it stores messages of which a destination is itself, the number of created and received messages, and delay of messages. Messages, which created in a node class, are sent to its transport class. Received messages are stored in its buffer. Delay of messages is calculated when a message or copy is reached in a node class.

A transport class is a basic class of transport layer protocols, such as UDP, TCP, and so on. While transport layer protocols are very important parts in legacy networks, they are not paid attention because end-to-end connections managed by transport layer protocols are not guaranteed in DTNs. Currently, we only implement an UDP, and it only forwards messages from a node class to a routing class.

A routing class is the most important part in DTNs since the performance of DTNs depends on routing protocols. Recently, many kinds of routing protocols, such as epidemic [14], spray and wait [13], PRoPHET [12], MaxProp [5], and so on, are proposed by researchers. We have implemented a represented routing protocol, epidemic. It is a basic flooding routing protocol, so it is extended to other routing protocols easily. We have defined a framework for routing protocols and implemented basic functions for the epidemic routing. For the epidemic routing, it manages a buffer of which drop policy is "drop oldest". In a routing class, it stores link bandwidth for calculating transmission time.

3.4.1 How to Add New Protocols

For add new transport and routing protocols in the our simulator, new protocols have to be based on our implemented frameworks. We have implemented frameworks for a transport layer protocol and a routing protocol. In frameworks, we have made function declarations for necessary functionalities in a transport layer protocol and routing protocol.

A transport layer protocol framework consists of declarations of basic functions, which are sending to and receiving from lower and upper layer. In UDP, which is implemented in our simulator, we have only implemented forwarding functions simply. For implementing new protocols, these forwarding functions must be implemented first, and next, other necessary functions should be implemented.

A routing class framework is also composed of declarations of basic functions, which are sending to and receiving from upper layer and other nodes' routing class, and buffer managements. For implementing new protocols, these functions should be implemented first, and other needed functions must be implemented. For example, we have implemented the epidemic routing protocol. For sending messages to upper layer, it only sends messages to upper layer if a messages' destination is only itself. When exchanging messages to another node, it first check whether receiving is rightly processed, and copies messages in own buffer. When messages are duplicated in own buffer, routing protocols must determine what message is dropped in its buffer if buffer is full. A drop policy is also determined for new routing protocols.

4. SIMULATOR VALIDATION AND PERFORMANCE EVALUATION

The simulator's accuracy is measured that simulation results is how close to the real systems. For validating our simulator, first, we compare simulation results with analytical results. In [9, 15], they proposed analytical models for epidemic routing. In [9], they developed the model for message delay in ad-hoc networks, and the inter-meeting time distribution model for random direction (RD) and random waypoint (RW) mobility models. In [15], they modeled message delay for various schemes using epidemic routing.

4.1 Simulator Validation

To validate our simulator, we compare simulation results with [9, 15]. We first substantiate the accuracy of contact event generation for RW mobility model, which is compared with [9]. Next, we compare the message delay of our simulator with that of analytical model in [15].

4.1.1 Accuracy of Contact Event Generation

Accuracy of contact event generation is the most important feature in our simulator. For validating it, we compare our results with analytical results in [9]. In [9], the inter-meeting rate λ_{RW} of two nodes is defined as follow,

$$\lambda_{RW} \approx \frac{2wrE[V^*]}{L^2}, \tag{1}$$

where $w = 1.3683$ is a constant to the RW model and $E[V^*]$ is the average relative speed between two nodes. Where $v = v_{min} = v_{max}$, there is $\lambda_{RW} \approx \frac{8wrv}{\pi L^2}$. We easily derive the total number of contacts from 1 as follow,

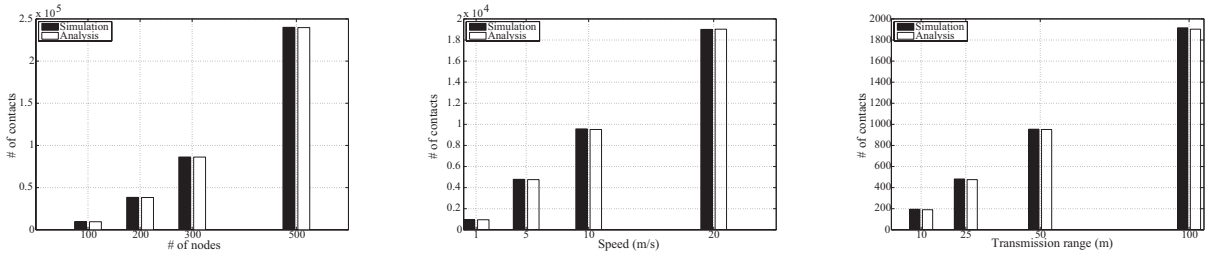
$$C = \lambda_{RW} \times \binom{N}{2} \times T = \frac{4wrvN(N-1)T}{\pi L^2}, \tag{2}$$

where N is the number of nodes in the scenario and T is the simulation time.

We simulated various speed and the number of nodes. We have 100, 200, 300, and 500 nodes which move in a map of 3,000x3,000m. Node speed variation is 1, 5, 10, and 20 m/s (respectively, 3.6, 18, 36, 72 Km/h). The variation of transmission rage is 10, 25, 50, and 100 meter. The mobility model is RW. We have simulated during 10,000 seconds. We created 10 different nodes movements with the same configuration, and took the average of simulation results.

Figure 3 shows simulations and analytical results. In each plot, X axis is represented the number of nodes, speed, and transmission range, respectively. Y axis is the total number

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650



(a) Number of contacts with various number of nodes (b) Number of contacts with various speed (c) Number of contacts with various transmission range

Figure 3: Comparing simulation results with analytical results

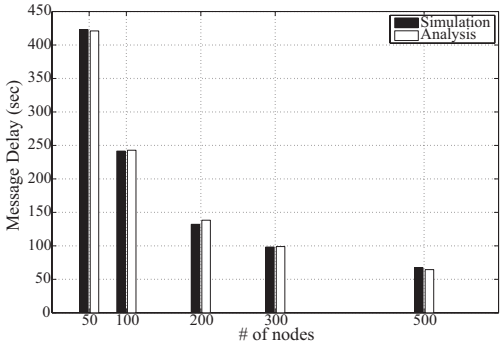


Figure 4: The message delay

of contact events in simulation time. Figure 3 is shown that the simulator accurately generates contact events.

4.1.2 Message Delay

Next, we validate the performance of routing protocols implemented by ours. In [15], the expectation of message delay, $E[T_d]$, in epidemic routing is modeled as follow,

$$E[T_d] = \frac{ln(N)}{\lambda_{RW}(N-1)}, \quad (3)$$

where N is the number nodes. It is the result with infinite bandwidth and infinite buffer.

In the simulation, we have 50, 100, 200, 300, and 500 nodes which move in a map of 3,000x3,000m. A node speed is 10m/s, and a transmission range is 50m. The mobility model is RW and buffer size is 50. We executed 10 times of simulation with different node movement scenarios and averaged them.

Simulation results are shown in Figure 4. In Figure 4, X axis represents the number of nodes, and Y axis is the message delay. As shown Figure 4, simulation results are the almost same as analytical results. It is that our implementation of epidemic routing protocol and message transmission functions are worked well.

4.2 Performance Observations

Our simulator offers frameworks for developing DTN protocols. Its performance mainly depends on developed protocols' computation and memory usages. Basically, simulator's performance depends upon the number of events, which is determined by simulation map size, the number of nodes, node's transmission range, node's speed, mobility models, and traffic generations. Also, it is dependent on the simulating computer environments.

For testing performance of simulator, we executed simulations in the linux PC (redhat 9.0) with Intel(R) Pentium(R) 4 CPU 3.20GHz and 2GBytes memory. We set java maximum heap size to 1024MBytes. Simulations are executed 100, 200, 300, and 500 nodes moving in 3,000x3,000m map. Node's speed is 10m/s, and node's transmission range is 50m. We used RW model for nodes' mobility. The bandwidth of link is infinite and buffer size of node is 50. The simulation time is 10,000 seconds. Even the most complex scenario is simulated, it only takes about 50 seconds and uses about 28.5MBytes memory. We think our simulator is executed in a reasonable time when more complex simulation scenarios are simulated.

5. FUTURE WORKS

In order for our simulator to become efficient, we have to implement more features, such as other mobility models, and routing protocols, and optimize simulator's performance. We also have to determine a result format to be shown easily.

Currently, we have only implemented RW mobility model. As describe above, there are many mobility models to represent the movement of mobile users. While RW mobility model is the most popular and used model in DTNs or mobile ad-hoc networks, it has already known shortcomings, and there are more realistic mobility models, such as map-based mobility model [6] or working day movement model [7]. More mobility models are described in [4]. In the mobility generator, it only covers models that the path of node during an interval is a straight line. In some mobility models, however, the path of node may be not a straight line. Therefore, we will modify our mobility generator more flexibly, and we will implement other mobility models.

As mentioned above, there are many routing protocols in DTNs, for example, spray and wait [13], PRoPHET [12], MaxProp [5], and so on. We have only implemented epidemic routing protocol which is the basic and most popular routing protocol in DTNs. Because other mentioned protocols are also popular routing protocols in DTN, and there are many researches are advancing based on these protocols, we will implement these routing protocols as soon.

The most popular network simulator ns-2 [1] presents gen-

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650

eral outputs with the pre-defined formant, and researchers get the own results by parsing outputs generated by ns-2. Similarly, our goal is that our simulator offers general outputs with pre-defined format. So we will define an output format, and implement functions offering results with this format.

Finally, we have to optimize our simulator's performance. While we have shown the performance of our simulator in Section 4.2, it is not guaranteed that the performance of our simulator is optimized. By simulating various scenarios and many times, we will further test our simulator's performance, and fix finding bugs. Also, we will be able to change codes with more efficient algorithms. By releasing our simulator, we will be able to receive some bug reports or fixed codes from other researchers using our simulator. We will also apply these features in our simulator.

6. CONCLUSION

In this paper, we have described a new event-driven simulator, which efficiently simulates Delay/Disruption Tolerant Networks (DTNs) protocols. It generates nodes' mobility models and generates contact events and message creation events, and these events are scheduled in order of time. Currently, we have implemented RW mobility model for nodes' movement generation, and epidemic routing protocol for routing. We have offered frameworks for developing other mobility models or DTN protocols. We have shown that our simulator generates contact events compared with the analytical model of RW mobility model and simulates epidemic routing protocol comparing the message delay with analytical model of epidemic routing, accurately. We also present future works for our simulator. We expect that our simulator will be useful to anyone investigating DTN protocols.

Acknowledgments

This work was supported by Korea Research Foundation grant funded by the Korea government (MOST) (No. 313-2008-2-D00883).

7. REFERENCES

[1] The Network Simulator NS-2.

[2] www.dtnrg.org/wiki.

[3] www.illuvatar.nu/ns-dtn/code/.

[4] Fan Bai and Ahmed Helmy. *Wireless Adhoc and Sensor Networks, Chapter 1*. Kluwer Academic Publishers, 2006.

[5] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *In Proc. IEEE INFOCOM*, 2006.

[6] V. Davies. Evaluating mobility models within an ad hoc network. Master's thesis, Colorado School of Mines, 2000.

[7] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40, New York, NY, USA, 2008. ACM.

[8] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[9] Robin Groenevelt, Philippe Nain, and Ger Koole. The message delay in mobile ad hoc networks. *Perform. Eval.*, 62(1-4):210–228, 2005.

[10] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158, New York, NY, USA, 2004. ACM.

[11] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[12] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[13] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, New York, NY, USA, 2005. ACM.

[14] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks, 2000.

[15] Ellen Zhang, Giovanni Neglia, Jim Kurose, and Don Towsle. Performance modeling of epidemic routing. Technical report, Dept. of Computer Science, University of Massachusetts, Amherst.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8650
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650