Real-time simulations of Mobile Ad-hoc Networks (MANET) in Opnet Modeler

H.T. Vu, M. Thoppian, A. Mehdian, S. Venkatesan, R. Prakash
The University of Texas at Dallas
Richardson, TX 75083
E-mail: {htv041000, mansi, axm056000, venky, ravip}@utdallas.edu
and
A.J.Anderson
Rockwell Collins, Inc

Abstract
The development of modern devices with wireless capability such as laptops, mobile phones, PDAs… makes the applications for Mobile Ad-hoc Networks (MANET) more promising. However, because of its "ad-hoc" property, it is not easy to study and predict the performance of the applications when implementing for such networks. Some systems have very strict requirements such as low latency, low delay and high-speed processing. One of the efficient ways to study these systems is to conduct real-time simulations. Real time simulations are also needed for hardware in the loop situations.

The question that arises is: is it possible to implement such real-time simulations in an efficient manner? In order to answer this question, we have implemented numerous simulations of MANET using OPNET as the primary tool with the goal of achieving real-time simulations while maintaining or increasing the fidelity. One of the primary techniques for achieving real-time results is to run the simulations concurrently in a multi-threaded system, which is supported by OPNET's parallel kernel. The results, however, show that the performance of the simulations is not necessarily improved and there are many reasons for this. In this paper, we describe the experiments, compare the simulation results, discuss the reasons why this happens and present cases in which the parallel technique does not work. We also propose some improvements which may help users in achieving better results for their simulations.

1. Introduction
A mobile ad-hoc network (MANET) consists of a number of mobile hosts such as laptops or personal devices with wireless radio interfaces dynamically forming a wireless network. Usually, a multi-hop routing protocol (such as AODV or DSR) is used to route the packet from one host to another in the MANET. Due to the "ad-hoc" property of MANET, a simulation tool is one of the best ways to experimentally measure the performances of their solutions for MANET. Many new and innovative developments in the telecommunication and network areas have taken advantage of simulation tools.

In the simulation literature, simulation is defined as "the discipline of designing a model of an actual or a theoretical physical system, executing the model on a digital computer, and analyzing the execution output" [1]. One of the main issues in MANET simulations is to compute the effects of the path of packets through a simulated network, which results in delays and loss of packets. Sometimes, as required by many critical military and commercial applications, a good simulation for MANET is needed not only to provide a flexible means for calculating delay and losses but also to ensure real time capabilities. A simulation is a real-time simulation if it takes no more time to finish than the actual run of the system. For example, if we implement the simulation for a scenario running for 1 hour and the simulation time is not more than one hour, then we can say it is a real-time simulation.

For a small size of network with a few tens of nodes, the total number of events is usually small and hence achieving the real time simulation is easy in most cases. However, when the size of the network is increases, it is not a trivial problem anymore. This is because, in general, the number of events is not linear in the number of nodes. Therefore, there is a need to implement advance techniques to speed-up the simulations, and parallelism is one such technique.

There are two approaches for implementing a simulation: sequential and parallel. In a sequential discrete event simulation, one maintains a global event list, a global clock and a single central processor for simulating the entire application. In this simulation, all the events are executed in the order of the timestamps which are assigned to them. As we can easily see, this technique is not efficient when the application becomes large and complicated, consisting of a very large number of events and computation. One the other hand, the use of parallel simulation intuitively improves the speed of the simulation by performing the scenario concurrently on multiple processors. In many cases, this technique helps the users to overcome the resource limitations and usually achieves speed-up significantly.

We have conducted several mobile ad hoc network simulation experiments to evaluate the speed-up achieved by using the OPNET Modeler 11.0 parallel kernel. Note that currently in OPNET Modeler only the MAC and PHY layers for wireless LAN, and not all the models, have been made parallel-safe. Our results show that the performance the OPNET parallel kernel for certain scenarios is not as good as what we expect. In the following sections we describe the conducted experiments and present some techniques that could potentially improve the performance of the simulations.

The rest of the paper is organized as follows. Section 2 discusses some work related to real-time simulations, real-time scheduling and parallel simulations tools. Section 3 presents and analyzes

the experiment results using OPNET Modeler. Section 4 proposes solutions to overcome the problems. Section 5 concludes the paper.

2. Related works

In [9] [10], the authors focus on solving the real-time problems in which, given a time bound, one needs to construct and execute a problem solving procedure that can produce a reasonable answer within the approximate time available. As for real-time simulations, efficiently managing and executing the events generated by the objects in the scenarios is very critical. Many scheduling techniques have been proposed for designing a real-time system or a real-time simulation tool [11] [12]. The main interest is to determine the schedule that defines when to execute which task to meet the deadline. Typical approaches to real-time scheduling assume that task priorities and resource needs are completely known in advance and are related to those of other tasks, so that a control component can schedule tasks based on their individual characteristics.

Several parallel simulation techniques have been proposed in the literature [2]. Among the simulation tools, some of them have employed parallel simulation techniques, such as OPNET [3], GloMoSim [4] and Qualnet [5].

3. Experiment results

In this section we conduct a set of experiments to analyze the performance of OPNET parallel kernel and to evaluate the use of OPNET parallel kernel to meet the requirements of real-time simulations. In our experiments we simulated small to large scale ad-hoc networks using both the sequential and parallel kernel. For parallel kernel, different values of window size were chosen. The goal of this experiments is to find out for what values of network size does the simulation execute in real-time. At the same time we play with the parameters of the simulation to explore what parameters play the important roles in archiving real-time simulation.

3.1 Project 1: 50 mobile nodes

We would like to run the simulation to see how different the result is between the parallel simulation and sequential simulation. As for the parallel simulation, we implement it in a system with two identical processors. We used the parallel kernel with various window size values, namely, zero second, 10^{-6} second and 10^{-7} second. The sequential simulation is implemented in the same system with sequential kernel running on one processor only.

In this project, we simulate 50 mobile nodes moving in a campus area of 1000m x 1000m. We virtually divide the network into several clusters which do not have communication between each other. We do this with the hope that the events in different clusters would be done concurrently, which obviously speedup the simulation. The routing protocol used in this project is AODV and the mobility model is Random Waypoint (default in OPNET).

In the first scenario we let 10 nodes (20% of total nodes) generate a constant stream of packet. We simulate this scenario in both parallel and sequential fashion for 1-hour duration. In the second scenario, we reduce the number of nodes that generate packets to only 5, which is 10% of the total number of nodes. The result in Figure 1 shows that there is a big difference in the number of events between the two scenarios. This result is reasonable because the higher number of nodes generating packets lead to higher number of events.

Since the number of nodes in the network is not very high, we expect that the elapsed time for both scenarios is not as long as the simulation, meaning that we can implement the scenario in real-time. This is because there is very little computational load on the nodes after receiving the packets. In our simulation experiment, the intermediate nodes just forward the packets and the destination nodes simply drop the packets after receiving them.

The result in figure 2 shows what we expected, that is, the elapsed time of both scenarios is not very high.

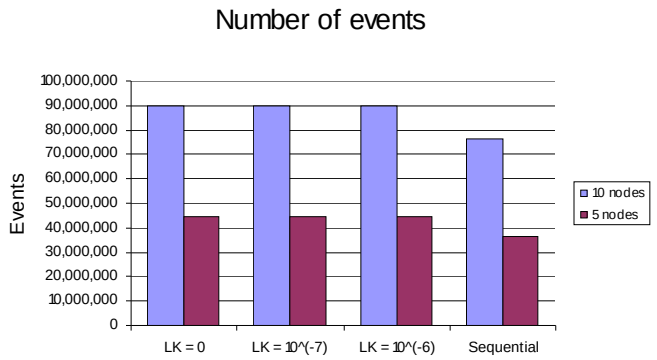

Figure 1: number of events generated

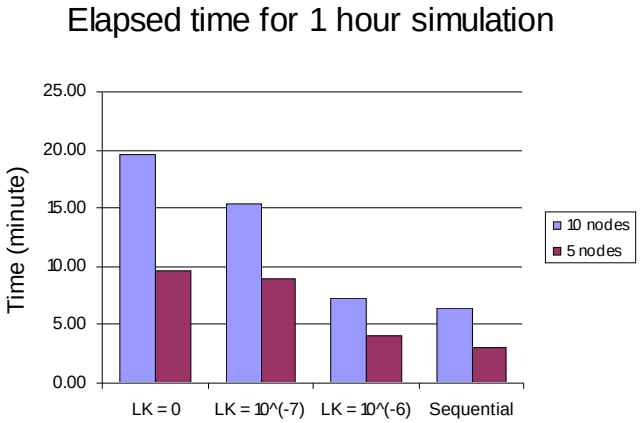

Figure 2: elapsed time for simulations

We observe that for both the scenarios the number of events generated by the parallel simulation kernel is a little higher (about 17%) than the number of events generated by the sequential simulation kernel. One of the reasons is the different random number generators used by the parallel and sequential simulation, which leads to the different number of events[1]. In addition, the elapsed time of the parallel simulation is surprisingly much longer than that of the sequential one. For

example, consider the scenario 1 with 10 nodes generating packets. The sequential kernel takes 6 minutes and 34 seconds to finish. With the same configuration, the parallel kernel with window size of zero takes 19 minutes and 53 seconds, which is about 3 times longer than that of the sequential one. When we modify the window size to 10^{-6} second and 10^{-7} second, the elapsed time is 15 minutes 44 seconds and 7 minutes 21 seconds, which is still 2.5 times and 10% higher than that of the sequential kernel time, respectively.

Besides, when implementing the sequential simulation, we generated the estimated speed up report to see how much speed-up can be achieved if the simulation was conducted using parallel kernel. The results are shown below:

Estimated speedup	
CPUs	*Speedup*
2	1.43
3	1.63
4	1.73
8	1.90
16	1.96
64	1.59

As we can see from the above results, the simulation can be theoretically speeded up by around 43% if we implement the scenario with parallel kernel on a 2-CPU system. However, in reality, it does not improve the performance. Because of this, our concern is to find out the reasons why the parallel simulation takes longer time to finish.

One of the reasons for the longer elapsed time is the parallel execution overhead required by the kernel to synchronize the threads resulting in many context switches and guarantee the consistency among program states. Another reason is because the 802.11 protocol, which we are using for the MANET simulation, inherently tries to avoid simultaneous events from happening[2].

However, in both simulations, the performance is still somewhat acceptable because the result of this experiment shows that the real-time constraint has been satisfied.

In the following simulations, however, we face some issues when increasing the scalability level of the network.

3.2 Project 2: 100 mobile nodes

Similar to the setting in the previous project, for this project we design the simulation in a campus area of 1500m x 1500m with 100 mobile nodes moving around. The mobile nodes are also grouped into clusters for increasing the possibility of having simultaneous events happening in different parts of the network. In each cluster, some nodes are selected to generate packet streams and send the packets to other nodes in the network. The routing protocol using in this project is AODV and the mobility model is Random Waypoint.

Obviously, the number of events and the elapsed time of this project should be higher than that of the former one; however what we want to know is whether the performance remains acceptable for larger sized network when the number of nodes is rising. Specifically, we want to see whether the real-time constraint is still satisfied or not.

For the two scenarios of this project, the ratio of sources node that generate packets remains the same, say 10% (10 nodes) and 20% (20 nodes). We keep other parameters of the simulation the same.

Surprisingly, the number of events in this scenario is much higher than the case when there are only 50 nodes. In fact, when we have only 10 nodes, the number of events is about 200,000,000 events, which is 3 times higher than that of the scenario with 50 nodes. The worst case happened when we let 20 nodes generate packet and the number of events increase substantially to about 600,000,000, which is 6.5 times higher than that of the former scenario. Hence, the number of events is not linearly increased with the number of nodes, as we expected. This result is shown in figure 3.

As a result of the increasing number of event, the time to finish the simulation does not guarantee to be real-time, as shown in figure 4. Once again, the elapsed time for parallel simulation is higher than that of the sequential. As for parallel kernel, it takes 40 minutes to finish scenario 1, which is still in the time bound but the scenario 2 requires 80 minutes to complete, and hence it does not satisfy the real-time constraints.

There is an issue that we would like to discuss here. In project 1, the parallel kernel produces different values when we adjust the window size. We try various window size values and what we observed shows that the value of 10^{-6} and 10^{-7} give the best performance in term of the number of events and the elapsed time. Hence, we expect that we could get the similar result in the second scenario. Yet it turns out that we were not able to implement the simulation with 10^{-6} and 10^{-7} window size values. Therefore, we do not have the results with window size other than zero to compare with the former project.

In fact, there is no optimized value of window size that has been confirmed, but in most of the cases, the above values of window size (10^{-6} and 10^{-7}) give us the best performance for the simulations. In addition, these values are the recommended values for 802.11 protocols[3], mainly because it is suitable with the propagation delay of 802.11.

In reality, the ad-hoc networks usually consist of more than 100 nodes. But our results show that we cannot always guarantee real-time constraint when implementing the simulation for such networks. Therefore, in order to speed up the ad-hoc network simulation, sometimes we need to sacrifice the fidelity level of the network, such as recording less number of statistics or reducing the packet processing.

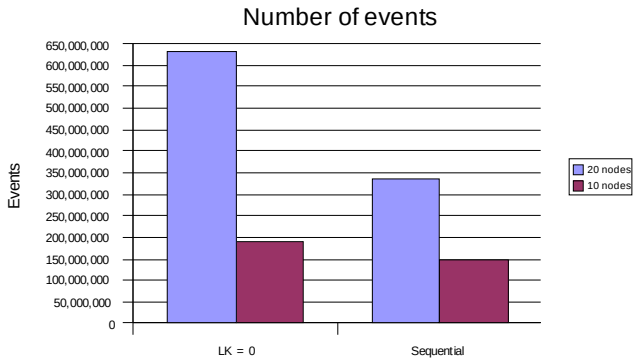

Figure 3: number of events generated

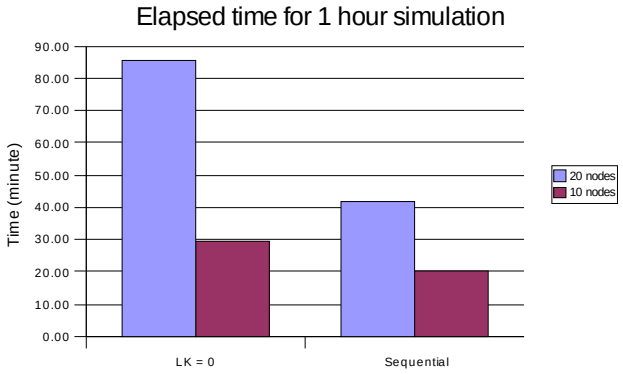

Figure 4: elapsed time of the simulations

4. Solutions

In our experiments, we observe that by increasing the value of window size, we could reduce the simulation run-time for parallel simulation. However, very large value of window size could lead to incorrect simulation execution. In OPNET Modeler, the window size is a static parameter and cannot be changed once the simulation is started. Hence, to ensure the correctness of simulation, it becomes necessary to choose the window size value in a conservative manner.

By making this window size value a dynamic parameter, during the execution of simulation, one could change the value of window size based on the amount of available parallelism. The window size could be dynamically shrunk to ensure that two dependent events do not fall within the same window. Similarly the window size can be expanded to allow a large number of independent events to fall within the same window.

For example, initially the default value of the window size can be large. In a process model, whenever a timer is set, the window value can be shrunk to the value of the timer. On the expiry of the timer, if no other timers are pending, one could increase the value of the window size to its default value.

Another possible solution is that the ad-hoc modules such as 802.11 protocols can be carefully re-designed to support parallel-safe code. This way the simulation will be improved significantly because the overhead caused by the parallel simulation can be avoided naturally in the kernel.

5. Conclusions

In our research we implemented real-time simulation for mobile ad-hoc networks. One of the techniques to achieve the desirable time constraints is to implement the simulation using parallel technique. We examine the potential of this technique using OPNET Modeler as the main simulation tool. The results show that in some cases the parallel technique does not really improve the performance of the simulation.

There are some reasons which make the parallel simulation not a ultimate and perfect technique to achieve the real-time constraints. The nature of 802.11 networks may limit the performance gain since they have not been made parallel-safe. The static window size is also a problem, because increasing the window size might improve the performance of the simulations, however in order to ensure the correctness of the simulations sometimes one need to reduce the window size so that two dependent events do not fall within the same window.

References

[1] P. A. Fishwick. Simulation Model Design and Execution: Building Digital Worlds. Prentice Hall, Inc., 1994.

[2] Fujimoto, R.M. Parallel discrete-event simulation. In Communication of the ACM, October 1990.

[3] Opnet Modeler Documentation opnet.com/support

[4] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: A Scalable Network Simulation Environment," Technical Report 990027, UCLA Computer Science Department, 1999.

[5] Qualnet Parallel Developer Documentation scalable-networks.com

[6] R. Meyer and R. Bagrodia. Path Lookahead: A Data Flow View of PDES Models. In Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS '99), 1999.

[7] Zhengrong Ji, Junlan Zhou, Mineo Takai, Jay Martin, Rajive Bagrodia. Optimizing Parallel Execution of Detailed Wireless Network Simulation. In 18th Workshop on Parallel and Distributed Simulation (PADS '04), 2004.

[8] A. J. Garvey, and V. R. Lesser. Design to-time real-time scheduling, IEEE Trans. System Man Cybern, Vol. 23, No. 6, Pages 1491-1502, 1993.

[9] B. D'Ambrosio. Resource bounded-agents in an uncertain world. Proc. IJCAI, Morgan Kaufmann Publisher Inc., San Francisco, CA, 1989.

[10] R. E. Korf. Depth-limited search for real-time problem solving. Real-time System, Vol. 2, No. 1-2, Pages 7-24, 1990.

[11] K. Ramamritham and J. A. Stankovic. Dynamic task scheduling in distributed hard real-time systems. IEEE Software, Vol.1, No. 3, Pages 65-75, 1984.

[12] J. A. Stankovic, K. Ramamritham and S. Cheng. Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. IEEE Trans. Computation, Vol. 34, No. 12, Pages 1130-1143, 1985.

1,2,3 The reasons for increase number of events and longer simulation time have been confirmed by e-mail communication with OPNET's technical support team.