Micro-benchmarking NS-2 and NS-3 Network Simulators Using the Terrain Aware Radio Propagation Extension

Sonja Stojanova[1], Leonid Djinevski[2], Igor Mishkovski [1], Sonja Filiposka[1], Dimitar Trajanov[1]

[1] Ss. Cyril and Methodius University – Skopje / CSE, Skopje, Macedonia

[2] Fon University, Skopje, Macedonia

Abstract - *This paper presents an overview of few of the most used network simulators in academia. Also, computational performance is evaluated for a radio-wave propagation model in simulation scenarios for one of the most popular and widely used network simulators NS-2, and its new significantly different version NS-3. Additionally we present the differences between both simulators, and few major issues that need to be taken into account when porting a NS-2 model to NS-3, from a developer's point of view. The results present a significant computational performance increase when using the NS-3 network simulator.*

1. INTRODUCTION

With portable radio equipment smaller, cheaper and more reliable, mobile radio communications industry has improved significantly during the last decade [1]. These devices have made a large scale deployment of easy-to-use radio communication networks, which created a trend of a demand for even greater freedom in the way people use the wireless communication networks [2].

This trend has also raised the demand in wireless ad hoc networks. A mobile wireless ad hoc network (MANET) is an infrastructure-less network that can be established anywhere on the fly [3]. This independence of infrastructure has made wireless ad hoc networks find many applications in real life scenarios like: rescue teams on crash sites, vehicle to vehicle networks, lumber activities, portable headquarters, late notice business meetings, military missions, and so on.

However, deployment of wireless ad hoc networks (no matter the low cost equipment) when evaluating large-scale networks, is not practical and time consuming. Therefore, wireless network simulation is of crucial importance when it comes to investigating and developing novel wireless ad hoc networking protocols.

There are many network simulators that were developed over the years. Most of them are an open-source community driven tool, while others are commercially supported. However, it is common for all that input scripts or commands are required to describe the state of a network and the events within, while the output is usually a trace of the simulation which documents every event that has occurs.

When simulating wireless networks, one of the most important parts is the radio-wave propagation, which provides information about the power of the signal when is received by the receiver node. There are two basic models that are usually supported in all network simulators, which are the free space propagation model and the ground reflection (two-ray) propagation model. However these models are terrain unaware, therefore unsuitable for wireless network simulation. In [4] we have developed a terrain aware radio-wave propagation model, based on Durkin's [5], which is an extension of the NS-2 network simulator [6] and brings wireless network simulation few steps closer to more realistic scenarios.

In this paper we present few of the most popular and widely used network simulators. We focus more on the NS-2 and the differences in NS-3. Additionally, we discuss the process of porting the NS-2 terrain aware radio-wave propagation model to NS-3 based on the Durkin's propagation model. This involves modification in the source of the model to account for many differences between the network simulators that are arising.

The rest of this paper is organized as follows: We present few of the most popular network simulators in Section 2. Durkin's radio-wave propagation model is presented in Section 3, followed by description of the applied testing methodologies used for micro-benchmarking NS-3 network simulator with our terrain aware radio-wave propagation extension in Section 4. NS-3 implementation of the Durkin's model is discussed in Section 5. The obtained results are analyzed in Section 6. A conclusion is provided in Section 7.

2. NETWORK SIMULATORS

In this section we review some of the most popular and widely used network simulators. More particularly, we are interested in the network simulators that are presently used in academia and in the industry. Prominent examples include ns-2, and its successor ns-3, OMNET++[7], the Java-based JiST [8], the process-oriented discrete-event simulator SimPY [9] and commercial tools such as OPNet modeler [10]. In addition to these environments, the WSN simulator TOSSIM [11] serves for dedicated research domains. However, ns-3, OMNET++ and JiST are gaining more and more acceptance in comparison with the long-established ns-2.

In order to model the behavior of the simulation nodes in a given network, all the network simulation in ns-2 are composed of C++ code and oTcl scripts that control the simulation, network topology and other network aspects. In ns-3 network simulations can be implemented in pure C++, whereas part of the simulation can be done optionally in Python as well. Moreover, ns-3 integrates architectural concepts and code from GTNetS [12], a simulator with good scalability characteristics. In contrast to ns-2 and ns-3, OMNET++ is a general purpose discrete

event-based simulation framework. In addition, OMNET++ contains the INET package which provides comprehensive collection of Internet protocol models. A particular protocol in OMNET++ is realized with simple modules and a host node is implemented by combining and linking these modules into a compound module. The simple modules in OMNET++ are written in C++, whereas the compound modules and the network simulations take place in NED, the network simulation language of OMNET++. In JiST the network simulations are implemented in Java. Network elements in JiST are represented by entities and the simulation events are formed by invoking methods between the entities, which are carried out in simulation time. This facilitates the paralellel execution of code at different entities. The main problem with JiST is the stagnation of the official development. Unlike the other simulators, no public available network models exist for SymPy. SimPy is a bare simulation API written in Python where the simulation entities are processes which are executed in parallel and can exchange Python objects among each other. Besides process abstraction and the exchange of objects, SimPy provides instructions for the synchronization of process simulation and commands for monitoring of the simulation data. For more detailed comparison of the aforementioned network simulators can be found in [13].

A. NS-2 Network Simulator

The NS-2 Simulator [6] is widely used in the networking research community and has found large acceptance as a tool to experiment new ideas, protocols and distributed algorithms. It is a discrete event driven sequential network simulator, developed at UC Berkeley by numbers of different researchers and institutions. NS-2 is suitable for simulating and analyzing either wired or wireless network sand is used mostly for small scale simulations. NS-2 is written in C++ and OTcl. The users define the network topology structure, the nodes, protocols and transmitting times in an OTcl script. The open source model of NS-2 encourages many researchers from institutions and universities to participate and contribute to improve and extend the project. NS-2 plays an important role especially in the research community of mobile ad hoc networks, being a sort of reference simulator [14]. Adding new network objects, protocols and agents requires creation of new classes in C++ and then linking them with the corresponding OTcl objects. The process of implementing a simulation is depicted in Fig. 1.

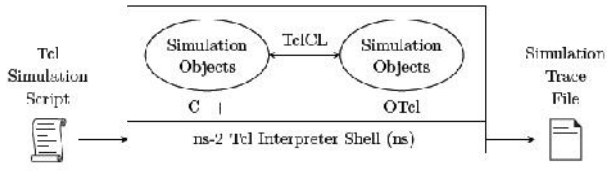

Figure 1. NS-2 process of implementing a simulation.

B. Differences in NS-3 Network Simulator

T NS3 network simulator started as an open-source project in 2006 [14], designed to replace the current popular NS2. Authors in [15] have performed a performance analysis of the computational and memory requirement of few major network simulators, where NS-3 was evaluated as one of the fastest network simulators.

Similar to NS-2, NS-3 is also a discrete event simulator, and relies mainly on C++ not just for the implementation of the simulation models as it is the case with NS-2, but for the overall system. The simulation modeling in NS-3 also differs from NS-2, such that scripting language oTcl is no longer supported, thus replaced with pure C++ and optional scripting language Python for setting up simulations [16]. This architectural change makes the NS-3 not so popular simulator, because it is a completely new simulator that has no backward compatibility for NS-2 legacy simulation models, thus there are very few models that are supported. On this other hand, NS-3 is superior when it comes to performance, easiness to implement models, and contributions maintenance even when authors have lost interest in their models.

Regarding memory management, where in NS-2 is completely manual, although NS-3 still supports memory management functions like new, delete, malloc and others, there are some classes that can automate the procedure [16]. An overview of the NS-3 internal organization is presented in Fig. 2.

Additionally in NS-3, modern features of C++ like standard template library (STL) [17] are supported. With the recent trends of parallel and distributed computing, Message Passing Interface (MPI) [18] is incorporated in some parts of NS-3. Additional attempts for utilization of GPUs are made for some model [19] [20], which yields that GPU support should be available soon. These technologies should enable NS-3 simulator to scale better when simulating large-scale networks.

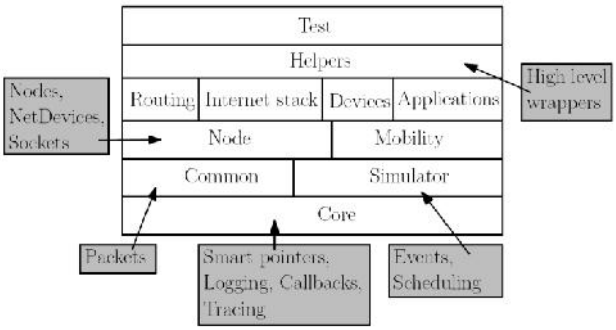

Figure 2. Overview of the NS-3 internal organization.

3. DURKINS RADIO PROPAGATION MODEL

Compared to free space and two-ray (ground reflection) models, the Durkin's model enhances the radio-wave propagation by introducing diffraction. Diffraction allows radio signals to propagate around the curved surface of the earth and propagate behind obstructions, however, it also causes radio signal distortions. These distortions are sufficient enough for useful signal to exist a region that is shadowed (no Line-of-Sight).

In mobile communication systems, diffraction loss occurs from the blockage of secondary waves such that only a portion of the energy is diffracted around an obstacle. That is, an obstruction causes a blockage of energy from some of the Fresnel zones, thus allowing only some of the transmitted energy to reach the receiver. Depending on the geometry of the obstruction, the received energy will be a

vector sum of the energy contribution from all unobstructed Fresnel zones.

In general, as shown on Fig. 3, if an obstruction does not block the volume contained within the first Fresnel zone, then the diffraction loss will be minimal, and diffraction effects may be neglected. In fact, as long as 60% of the first Fresnel zone is kept clear, the further Fresnel zone clearance does not significantly alter the diffraction loss.

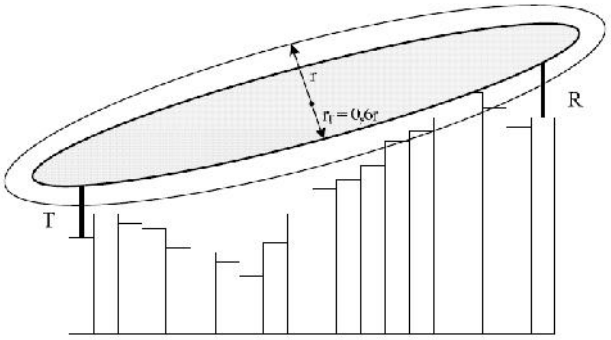

Figure 3. Fresnel zones and diffraction loss due to zone blockage.

When shadowing is caused by a single object such as a hill or mountain, the attenuation caused by diffraction can be estimated by treating the obstruction as a diffracting knife edge. This is the simplest of diffraction models, and the diffraction loss in this case can be readily estimated using the classical Fresnel solution for the field behind a knife edge.

The diffraction gain due to the presence of a knife edge, as compared to the free space, is given by.

$$G_d(dB) = 20log|F(v)|. \quad (1)$$

where F(v) is the Fresnel integral, a function of the Fresnel-Kirchoff diffraction parameter v, defined as:

$$v = h\sqrt{\frac{2(d_1+d_2)}{\lambda d_1 d_2}}. \quad (2)$$

where h is the relative height of the obstruction. In practice, graphical or numerical solutions are relied upon to compute the diffraction gain. An approximate solution for (1) provided by [21] is:

TABLE I. TYPE SIZE FOR CAMERA-READY PAPERS

The terrain presented in DEM file format	▪ 1000m x 1000m ▪ 1:1:0.1 resolution ▪ highest relative point at 200m
Nodes	▪ 100 uniformly dispersed ▪ Transmision range 250m (IEEE 802.11b standard equipment)
Nodes mobility	▪ Random direction model in terrain boundries ▪ node speed: 0, 1, 2, or 5m/s with derivation of 0.1m/s
Antenna	▪ 1.5m (no relative offset of the node)
Route discovery and path	AODV protocol [24]
Network load	0.7 – 7Mbps using UDP data packets of 1KB
Simulation time	2.5 hours (average battery life of a notebook)

$$\begin{aligned} G_d(dB) &= 0 \\ G_d(dB) &= 20\log(0.5 - 0.62v) \\ G_d(dB) &= 20\log(0.5e^{-0.95v}) \end{aligned} \quad (3)$$

$$G_d(dB) = 20\log\left(0.4 - \sqrt{0.1184 - (0.38 - 0.1v)^2}\right)$$
$$G_d(dB) = 20log\left(\frac{0.225}{v}\right)$$

4. PORTING THE DURKIN'S MODEL TO NS-3

Porting a NS-2 model to NS-3 is not a straight forward procedure. There are many issues that need to be taken into account.

There are few mobility generator models that determine the position and velocity of the nodes during a simulation. Unlike NS-2 where mobility is based on the vector velocity model, the NS-3 models maintain the Cartesian position and speed of the nodes. Since, NS-3 is still under development, in [22] authors have found discrepancy of the node positions of several magnitudes, we decided to use our own mobility generator in our simulation. However, the NS-2 mobility format requires major revision and retrofitting to accommodate to the NS-3 format. The same is required for the traffic generation. Other model revisions were performed for the packet and header architecture which is slightly different in NS-3, as well as DEM [23] terrain modeling.

5. TESTING METHODOLOGIES

In order to determine the computation performance of our Durkin's radio propagation model simulations, we investigated several sets of scenarios. Ad hoc network performances evaluation was the main focus for the chosen scenarios. Execution time was obtained for each parameterized simulation.

The terrain details, node settings, node mobility, discovery protocols, network load, and simulation settings for the simulations are presented on Table I.

6. RESULTS

This section presents the results of the experiments on NS-2 and NS-3 network simulators by varying simulation parameters as presented in Table I.

The obtained results from the traces of both NS-2 and NS-3 simulations were relatively identical as expected. We focus on the performance of the NS-2 and NS-3 simulators, which is the main goal of this paper.

The overall execution time of the simulation vs. the terrain factor is presented in Figure 4. Each curve represents the different mobility scenarios for node speed of 0, 1, 2, and 5 m/s for both network simulators NS-2 and NS-3. Varying the network load did not significantly influence the computation performance of the overall execution time of the simulation.

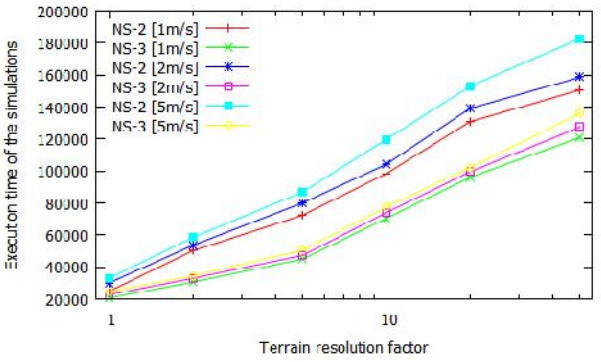

Figure 4. Overall execution time of the simulation vs. the terrain factor.

7. CONCLUSION

In this paper NS-3 implementation of the Durkin's terrain aware radio-wave propagation model is presented. Additionally, a computational performance analysis of the NS-2 and the NS-3 networks simulators as software tools is introduced.

The obtained results present a significant performance increase for the NS-3 simulation, which is based on the improved memory management and use of aggregation system which prevents unneeded parameters from being stored when they are unnecessary. Also the packets are much more lightweight. Additionally, the required computations are significantly smaller in NS-3, which is due to the absence of oTcl, since the oTcl scripting language introduces overhead.

In future work, we intend to find the bottleneck of our NS-3 implementation which are the most time consuming tasks in the propagation model, and try to optimize them by parallelization or find a way that is going to be more efficient. Additionally, we plan to extend our propagation model to different terrain representations.

REFERENCES

[1] Hekmat, R., *Ad-hoc Networks: Fundamental Properties and Network Topologies*, Springer 2006.

[2] Brown, J.S. and Duguid, P., *Social life of information*, Harvard Business School Press 2000.

[3] Tonguz, O.K. and. Ferrari, G., *Ad Hoc Wireless Networks: A Communication-Theoretic Perspective*, John Wiley & Sons 2006.

[4] Filiposka, S. and Trajanov, D., *Terrain Aware 3D Radio Propagation Model Extension for NS-2*, Transactions of the Society for Modeling and Simulation International, 2010.

[5] Edwards, R. and Durkin, J., *Computer Prediction of Service Area for VHF Mobile Radio Networks*, Proceedings of the IEEE, Vol. 116, No. 9, pp. 1493-1500, 1969.

[6] *NS-2 network simulator*, available at: http://nsnam.isi.edu/nsnam/index.php.

[7] Varga, A. and Hornig, R., *An overview of the OMNeT++ simulation environment*, In Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008'), March 2008.

[8] Barr, R., Haas, Z.J. and van Renesse, R., *JiST: an efficient approach to simulation using virtual machines*, Softw, Pract. Exper, 35(6):539–576, 2005.

[9] K. Mueller. SimPy documentation.

[10] *OPNET Technologies Inc*, OPNET modeler website, available at: http://www.opnet.com/solutions/network\ rd/modeler.html.

[11] Levis, P., Lee, N., Welsh, M. and Culler, D., *TOSSIM: accurate and scalable simulation of entire TinyOS applications*, In Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), 2003.

[12] Riley, G., *Large scale network simulations with GTNetS*, In Proceedings of the 2003 Winter Simulation Conference, 2003.

[13] Weingartner, E., vom Lehn, H. and Wehrle, K., *A performance comparison of recent network simulators*, Proc. IEEE Int. Conf. Commun., pp.1-5 2009.

[14] Henderson, T.R., Roy, S., Floyd, S. and Riley, G.F., *ns-3 project goals*, Proceeding from the 2006 workshop on ns-2: the IP network simulator, ACM, 2006.

[15] Weingartner, E., Vom Lehn, H. and Wehrle, K., *A performance comparison of recent network simulators*, IEEE International Conference on Communications, ICC'09, pp. 1–5, 2009.

[16] *NS-3 Tutorial*, Available at: http://www.nsnam.org/.

[17] Musser, D.R., Derge, G.J. and Saini, A., *STL tutorial and reference guide: C++ programming with the standard template library*, Addison-Wesley Professional 2009.

[18] Gropp, W., Lusk, E. and Skjellum, A., *Using MPI: portable parallel programming with the message passing interface*, MIT press 2009.

[19] Swenson, B. and Riley, G., *Simulating Large Topologies in ns-3 using BRITE and CUDA Driven Global Routing*, sixth International Conference on Simulation Tools and Techniques SIMUTools 2013, Cannes, France, March, 2013.

[20] Papanastasiou, S., Mittag, J., Strom, E.G. and Hartenstein, H., *Bridging the gap between physical layer emulation and network simulation*, Wireless Communications and Networking Conference (WCNC), IEEE 2010.

[21] Dadson, C. E., Durkin, J., Martin, E., *Computer Prediction of Field Strength in the Planning of Radio Systems*, IEEE Transactions on Vehicular Technology, Vol. VT-24, No. 1, pp. 1-7, 1975.

[22] Kamoltham, N., Nakorn, K.N. and Rojviboonchai, K., *From NS-2 to NS-3-Implementation and evaluation*, Computing, Communications and Applications Conference (ComComAp), pp. 35--40, IEEE 2012.

[23] Maune D.F., *Digital elevation model technologies and applications: the DEM user's manual*, American Society for Photogrammetry and Remote Sensing, 2007.

[24] Perkins, C., *Ad hoc On-Demand Distance Vector (AODV) Routing*, Internet-Draft Experimental RFC 3561, july 2003.