A Comparison of the Present MANET Simulators

***Roxana ZOICAN, Ph. D.*[*], Dan GALAȚCHI, Ph. D.*[*]**

Cuvinte cheie. *Comunicați radio, rețele ad-hoc, calitatea serviciilor.*

Rezumat: *Inginerii de rețea utilizează simulatoare pentru depanarea sistemelor şi testarea fiabilității şi a calității serviciilor oferite de hardware-ul de rețea şi de protocoalele de telecomunicații. Astfel, simularea devine un pas important în dezvoltarea rețelelor de comunicații fără fir. In scopul obținerii unor rezultate relevante în urma simulărilor, este important ca modelul pe baza căruia este realizat simulatorul, să fie conform realității. În această lucrare sunt prezentate rezultatele simulărilor obținute cu simulatoarele OPNET Modeler, NS-2, GloMoSim. De asemenea se prezintă un nou simulator pentru rețelele MANET şi se pun în evidentă diferențele dintre performanțele obținute cu aceste simulatoare.*

Keywords. *Wireless communications, ad-hoc networks, QoS.*

Abstract: *Network engineers use simulation tools to debug and test the reliability and QoS of network protocols as well as hardware equipment. This makes simulation a very important step towards the deployment of wireless communication networks. For the results of the simulation to be meaningful, it is important that the model on which is based the simulator matches as closely as possible the reality. In this paper there are presented the simulation results of a straightforward algorithm using several popular simulators (OPNET Modeler, NS-2, GloMoSim). It is then offered a simulation tool that it was developed in our university to aid future MANET simulation (MANSIM) .The results tend to show that significant divergences exist between the simulators.*

1. INTRODUCTION

Mobile Ad Hoc Networks (MANETS) are wireless mobile nodes that cooperatively form a network without infrastructure. In other words, ad hoc networking allows devices to create a network on demand without prior coordination or configuration. Thus, nodes within a MANET are involved in routing and forwarding information between neighbors. Because there is no coordination or configuration prior to setup of a MANET, there are several challenges.

These challenges include routing packets in an environment where the topology is changing frequently, wireless communications issues, and resource issues such as limited power and storage. The leading way to research solutions to these difficult MANET challenges is simulation. In this paper it was developed a list of simulation pitfalls that impact the credibility of a simulation-based study. Because the pitfalls impact different phases of a simulation-based study, the pitfalls have been grouped into the following categories: simulation setup, simulation execution, output analysis, and publishing.

A. Simulation Setup. Simulation setup is the phase of a MANET research effort that is most often

[*] Politehnica University of Bucharest, Electronic and Telecommunication Faculty, Telecommunication Department, Iuliu Maniu 1-3, 77202, Bucharest 6, E- Mail : roxana@elcom.pub.ro, dg@elcom.pub.ro

skipped or overlooked; and if the setup phase is done improperly, the credibility of the simulation study is flawed from the start. Setup begins with determining the simulation type and validating the model.

1) Simulation Type: Although selecting the type of simulation appears to be a trivial step, not identifying the type of simulation (terminating vs. steady-state) is a commonly overlooked step for researchers. Not determining the simulation type can lead to unrealistic simulations with statistically unsound results. The most common error made by researchers is to execute one type of simulation and apply the results to the other type of simulation. For example, executing a terminating simulation for a set number of seconds and claiming the results represent the steady-state behavior.

2) Model Validation and Verification: After the type of simulation is determined, the simulation model itself must be prepared. Many researchers download the NS-2 simulator, compile it and begin to execute simulations with a model that has not been validated in their environment. Likewise, the protocol that is being evaluated must be verified to ensure it has been coded correctly and operates in accordance with the protocol specifications. Not validating the model or verifying code is a common pitfall.

3) Variable Definition: NS-2 uses hundreds of configurable variables during an execution in order to meet its general wired and wireless network simulator requirements. There are 538 variables defined in the ns-default.tcl file of NS-2.1b7a and there are 674 variables defined in the ns-default.tcl file of NS-2.27. The large number of variables makes it difficult to track each variable's default setting. Additionally, an increase in the number of variables between the different NS-2 versions indicates there is a rising number of variables with each new version of NS-2.

B. Simulation Execution. Executing the simulation is where a lot of time is spent. Therefore, it is important to conduct the execution portion correctly. In this paper there were highlighted several execution pitfalls that have been discovered; these pitfalls impact data output, analysis, and ultimately results.

1) Setting the PRNG (Pseudo Random Number Generator) Seed: One mistake it has been seen in NS-2 based simulation studies concerns not setting the seed of the PRNG properly. NS-2 uses a default seed of 12345 for each simulation run. Thus, if an NS-2 user does not set the seed, each simulation will produce identical results. Additionally, if the seed is not set or is set poorly, it can negate the independent replication method which is typically used in analysis.

2) Scenario Initialization: Another pitfall is not initializing the scenario correctly. On terminating simulations, the network is usually started in a certain configuration that represents the start of the simulation window. For example, if the researcher is trying to simulate a protocol's response to a failure event, he needs to have the failure as the initialization of his analysis. Likewise, simulations often start with empty caches, queues, and tables. The simulation fills the caches, queues, and tables until a steady-state of activity is reached. Determining and reaching the steady-state level of activity for the queues, caches, and tables is part of the initialization.

After steady-state is reached, simulated data can be used for performance comparison.

3) Metric Collection: Another area of concern is the metric measurements collected during execution. If the simulation executes properly, but the researcher does not obtain the data he needs from the simulation, the simulation is worthless. Appropriate output is especially critical if output has to be correlated. For example, if the researcher is trying to track steady-state protocol performance after a cer-

tain configuration change, the researcher has to be able to identify the configuration change occurrence in the output.

4) Generating Sufficient Runs: A final pitfall with discrete-event simulation execution is not accounting for the model's innate randomness in the experiment. Executing the simulation once will produce results, maybe even good results. However, the single point estimate produced will not give the researcher sufficient confidence in the unknown mean.

2. The simulated algorithm

This paper presents a set of measures collected during the simulation of the flooding algorithm on different simulators (OPNET, NS-2, GloMoSim, MANSIM). This very simple broadcast algorithm (which simply consists in forwarding to the neighboring nodes every message received for the first time) is a basic building block for several wireless network protocols (such as routing or groups membership protocols, for example). It has been carefully implemented it in each simulator paying special attention in setting the same parameters and considering the same scenarios. But, surprisingly, there have been collected very different results, barely comparable.

The broadcast of messages is a frequently used operation to spread information to the whole network. It is the simplest building block used by network algorithms and is often required by higher level protocols such as most routing algorithms. For this reason, it is important for the broadcast to be implemented in the most efficient way. Its performance is likely to affect the global efficiency of any protocol using it.

For the simulations there were considered here, a peer-to-peer wireless network of several tens of mobile nodes (typically 100) randomly placed on a 5km x 5km area. The wireless network operates in the ad-hoc mode, without any central infrastructure (such as an access point). Every node (or peer) has the same possibilities and functionalities.

A rather direct and simple way to implement broadcast is to food the message over the network. When a node initiates a broadcast, it transmits the message to its neighborhood. By neighborhood, it is understood that all the nodes within the sender's transmission range. Then, when the message is received for the first time, the recipient rebroadcasts it. With ideal conditions (i.e. all node receive the broadcast) in a network of N nodes, a single broadcast will generate exactly N copies of itself which are likely to increase the probability of collisions. Moreover, most nodes will receive the same message several times keeping the shared medium unnecessarily busy.

3. Types of simulators

OPNET Modeler. OPNET Modeler is a powerful network simulator developed by OPNET. It can simulate all kinds of wired networks, and a 802.11 compliant MAC layer implementation is also provided. Although OPNET is rather intended for companies to diagnose or reorganize their network, it is possible to implement one's own algorithm by reusing a lot of existing components. Most part of the deployment is made through a hierarchical graphic user interface.

Basically, the deployment process goes through the following phases. First you have to choose and configure the node models (i.e. types) you want to use in the simulations - for example a wireless node, a workstation, a firewall, a router, a web server, etc. Then you build and organize your network by connecting the different entities. The last step consists in selecting the statistics you want to collect during the simulations. In the case presented here, it was

necessary to create a new node model which encapsulates the 802.11 MAC layer of OPNET and, on top of it, an application process that implements the flooding algorithm. A process model (such as the flooding algorithm) is described as a state machine. Each state can have some code that is executed when it gets active. A transition that links two states is followed whenever a certain condition carried by the transition is true.

The difficulty with OPNET Modeler is to build this state machine for each level of the protocol stack.

NS2. NS-2 is a discrete event network simulator that has begun in 1989 as a variant of the REAL network simulator. Most MANET routing protocols are available for NS-2, as well as a 802.11 MAC layer implementation.

NS-2's code source is split between C++ for its core engine and OTcl, an object oriented version of TCL for configuration and simulation scripts. The combination of the two languages offers an interesting compromise between performance and ease of use.

Implementation and simulation under NS-2 consists of 4 steps:

(1) implementing the protocol by adding a combination of C++ and OTcl code to NS-2's source base;

(2) describing the simulation in an OTcl script;

(3) running the simulation and

(4) analyzing the generated trace files.

Implementing a new protocol in NS-2 typically requires adding C++ code for the protocol's functionality, as well as updating key NS-2 OTcl configuration files in order for NS- 2 to recognize the new protocol and its default parameters. The C++ code also describes which parameters and methods are to be made available for OTcl scripting. The NS-2 architecture follows closely the OSI model.

Some disadvantages of NS-2 stem from its open source nature. First, documentation is often limited and out of date with the current release of the simulator. Fortunately most problems may be solved by consulting the highly dynamic newsgroups and browsing the source code. Then code consistency is lacking at times in the code base and across releases. Finally, there is a lack of tools to describe simulation scenarios and analyze or visualize simulation trace files. These tools are often written with scripting languages. The lack of generalized analysis tools may lead to different people measuring different values for the same metric names. The learning curve for NS-2 is steep and debugging is difficult due to the dual C++/OTcl nature of the simulator.

A more troublesome limitation of NS-2 is its large memory footprint and its lack of scalability as soon as simulations of a few hundred to a few thousand of nodes are undertaken.

GloMoSim. GloMoSim is a scalable simulation environment for wireless and wired networks systems developed initially at UCLA Computing Laboratory. It is designed using the parallel discrete-event simulation capability provided by a C-based parallel simulation language, Parsec. GloMoSim currently supports protocols for purely wireless networks. It is build using a layered approach. Standard APIs are used between the different layers. This allow the rapid integration of models developed at different layers by users.

To specify the network characteristics, the user has to define specific scenarios in text configuration files: app.conf and Config.in. The first contains the description of the traffic to generate (application type, bit rate, etc.) and the second contains the description of the remainder parameters. The statistics collected can be either textual or graphical. In addition, GloMoSim provides various applications

(CBR, ftp, telnet), transport protocols (tcp, udp), routing protocols (AODV, flooding) and mobility schemes (random waypoint, random drunken).

With GloMoSim, the difficulty was to describe a simple application that bypasses most OSI layers. The bypass of the protocol stack is not obvious to achieve as most applications usually lie on top of it. Compared to OPNET, for example, the architecture is much less flexible.

MANSIM. It was developed a simulation tool that can be used to increase the credibility of MANET simulations (i.e., scenario development, model validation, protocol verification, and results analysis). It is a visualization and animation tool

MANSIM, like SCORES (SCenariO characterizer for Simulation) tool, takes simulation input parameters, such as number of nodes, simulation area dimensions, and transmission range, and generates characterization data for the scenario. To improve the credibility of our simulation studies, the MANET community needs a way to characterize a simulation scenario in categories such as mobility, density, and coverage. The MANSIM tool is our initial effort into providing this characterization. It has substantially impacted our own MANET research efforts.

4. Simulation results

Several scenarios have been implemented on each simulator (OPNET, GloMoSim, NS-2 and MANETSIM). For the simulations it was considered a wireless ad-hoc network of 100 mobile nodes, uniformly placed on a terrain of 5 km x 5 km. All nodes runs the IEEE 802.11 MAC protocol in the ad-hoc mode (or peer-to-peer). In opposition to the infrastructure mode, the ad-hoc mode does not require any central base station (or access point). The physical layer has a data rate of 2,5 Mbps.

Each simulation run lasts 300 seconds during which 10 nodes initiate 50 broadcasts that will be flooded throughout the whole network.

Nodes' mobility is an important metric when evaluating ad-hoc networks and can be modeled in several ways. A lot of papers have compared and discussed the relevance of these models. It was chosen the random waypoint model. A node chooses a destination within the physical terrain. Then it moves in the direction of the destination at a constant speed uniformly chosen between $\mathbf{V_{min}}$ and $\mathbf{V_{max}}$. After it reaches its destination, the node stays there for PAUSE TIME time period.

In order to evaluate the performance of the flooding algorithm, there were defined three important metrics. The first one gives information about the time needed to flood a message. The second measures the general efficiency of the algorithm. And the last one stores the overhead of messages (unnecessarily) flooded in the network.

Time delay: For a node *n*, this is the average time needed for one packet to reach *n*.

Success rate: For a node *n*, this is the difference between the expected and the actual number of messages received at *n*.

Overhead: For a node *n*, this is the sum of duplicated packets received by *n*.

The three metrics above are all locally computed and then averaged among all the nodes. The conjunction of these metrics gives a pretty good idea of the performance of the flooding algorithm.

This part presents the results collected during the simulation of the flooding algorithm with the different simulators.

The first scenario (Fig. 1) studies a critical factor that influences the success rate in MANET: the effective transmission range. It can be noticed important differences between the simulators not

only in term of absolute value but also in terms of general trend.

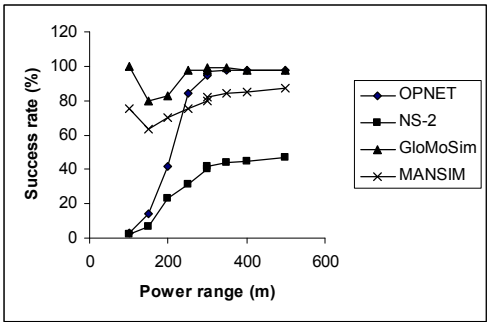


Fig. 1. Success rate vs. Power range.

The next scenario (Fig. 2) evaluates the effects of node mobility on the flooding's ability to deliver packets reliably. Here again, it can be noticed a significant difference in terms of success rate. The most important is the maximal success rate that can be achieved by NS-2.

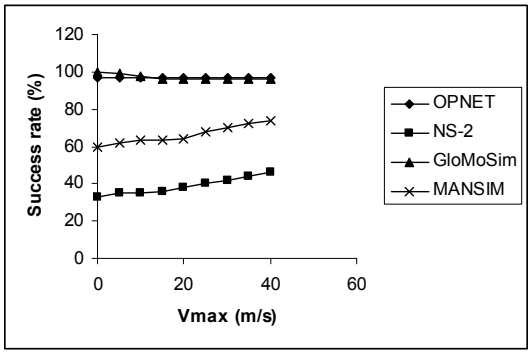


Fig. 2. Success rate vs. Mobility.

The third scenario (Fig. 3) presents the average overhead of messages flooded in the network for a single simulation run. This metric is related to the mean number of reachable neighbors (i.e. within transmission range). In this scenario, OPNET receives about three times more duplicate packets.

The last scenario (Fig. 4) compares the average time delay needed to flood a message throughout the whole network. This metric increases with the number of hops from the source to the destination and also whenever collisions occur. OPNET simulates a very high time delay compared to NS-2 GloMoSim and MANSIM

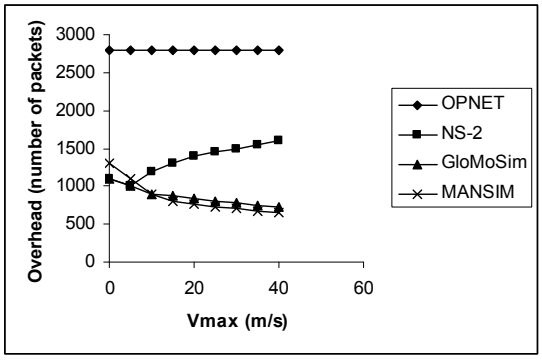


Fig. 3. Overhead vs. Mobility.

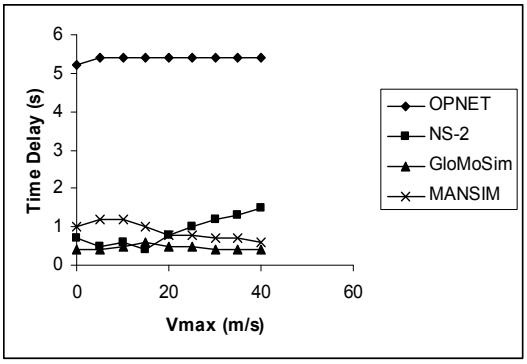


Fig. 4. Time Delay vs. Mobility.

Analysis and interpretation. The simulation results of the flooding algorithm shown in the above section demonstrate that the modelation of the MAC protocol and of the physical layer can lead to pretty different results depending on the simulator. People of the MANET community have rarely noticed these serious divergences because it is often too time costly to become familiar with more than one simulator as long as it is not necessary. Several reasons may help to explain these unexpected results. One major issue, from the physical layer point of view, is that the reality (i.e. the environment and mobility model) is hard to describe and the simulators usually provide too simple and too general wireless propagation

models. But even if it was possible to extract all environmental parameters from the reality, it would be anyway almost impossible to integrate them efficiently. Thus the physical layer implementation may significantly differ from one simulator to another simply because the levels of detail are not the same.

Each implementation relies on several abstractions and simplifications that may lead to different results. Then, the implementation of a new protocol itself is not straightforward and hard to transpose from one simulator to another.

Finally, through their successive releases each simulator announces the correction of newly discovered bugs especially in the lower layers such as the MAC and physical layers. It is reasonable to think that MANET simulators still contain errors or incompatibilities with respect to the IEEE 802.11 standard. These remarks tend to show that simulations, as such, do not sufficiently help to deploy and diagnose real networks as their results are not directly exploitable.

5. Conclusions

In this paper there have been presented the results of the simulation of the flooding algorithm using three popular mobile ad-hoc network simulators and also our MANSIM simulator. It was intentionally chosen one of the simplest algorithms in order to rely on a minimal set of building blocks.

Moreover, flooding is a wide used technique used to spread information in a whole network by wireless applications and protocols (e.g. routing). Through intensive simulations there have been noticed important divergences between the simulators and, in some cases, results that are barely comparable. The differences are not only quantitative (not the same absolute value) but also qualitative (not the same general behavior). This observation makes the simulation phase less credible as it is difficult to tell which simulator describes better the reality. Simulations do not really improve the development process. In conclusion, standalone simulations do not really fit the actual needs of wireless applications developers. Instead of simulations, it is probably more realistic to consider a hybrid approach in which only the lowest layers (MAC and physical) and the mobility model are simulated and all the upper layers (from transport to application) are executed on a dedicated hosts.

6. References

[1] **J. Heidemann, N. Bulusu, J. Elson, C.** Intanagonwiwat and K. Lan and Y. Xu and W. Ye and D.Estrin, and R.Govindan, “Effects of detail in wireless network simulation”, Proceedings of the SCS Multiconference on Distributed Simulation, pages 3-11, Phoenix, Arizona, USA, 2001.

[2] The network simulator - NS-2, http://www.isi.edu/nsnam/ns.

[3] **M. Takai, J. Martin, and R. Bagrodia,** “Effects of wireless physical layer modeling in mobile ad hoc networks. In MobiHoc 2001, 2001.