AD HOC NETWORK SIMULATORS FEATURES

S. Filiposka[1], I. Trendafilova[2], D. Trajanov[1], A. Grnarov[1]

[1)]e-Technology Center, Fac. Electr. Engineering, Sts. Cyril and Methodius University
Karpos 2 bb., PO Box 574, 1000 Skopje, R. Macedonia

[2)] Pivara Skopje, A.D. Skopje, ul. 808 br. 12 ,1000 Skopje, R. Macedonia

{mite, filipos, grnarov}@etf.ukim.edu.mk; irenat@paris.com

Abstract: The deployment of wireless applications or protocols in the context of MANETs, often requires stepping through a simulation phase. There are several popular simulators, such as OPNET Modeller, GloMoSim or NS-2. Since NS-2 is more frequently used in worldwide network societies we present our experiences gained with the use of NS-2 for simulations of wireless mobile ad hoc networks. Facing the problem of poor documentation for ad hoc networks, our research resulted in reach documentation for NS-2 trace files and behaviour of ad hoc networks. We observed the flexibility of NS-2 simulator, by the means of adding new application layers, change of existing routing protocols, and incorporating various extensions created outside the NS-2 development team.

Keywords: Mobile Ad-hoc networks, simulations, network simulators

1 Introduction

Mobile Ad-hoc NETworks (MANET) have become more and more popular, during the last ten years. Mobile hosts such as notebook computers are now easily affordable and are becoming quite common in everyday business and personal life. At the same time, network connectivity options for use with mobile hosts have increased dramatically, including support for a growing number of wireless networking products based on radio and infrared. With this type of mobile computing equipment, there is a natural desire and ability to share information between mobile users. In areas in which there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use, wireless mobile users may still be able to communicate through the formation of an ad hoc network.

An ad hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any centralized administration or standard support services. Therefore, the mobile nodes in the network dynamically establish routing among themselves to form their own network “on the fly” operating not only as a host but also as a router, forwarding packets for other mobile nodes in the network that may not be within direct wireless transmission range of each other.

Prior to the research in the field of wireless networks, a choice has to be made for a simulator that will best fulfill our needs, like: performance analysis of ad hoc networks, easy development of new protocols or update for the current ones. The deployment of wireless applications or protocols in the context of MANETs, often requires stepping through a simulation phase. For the results of the simulation to be meaningful, it is important that the model on which the simulator is based matches as closely as possible to reality. There are several popular simulators, such as OPNET Modeller[6], GloMoSim[5] or NS-2 [4]. Each of them provides advanced simulation environments for testing and debugging networking protocols, including wireless applications. After exploring the possibilities of each of the above-mentioned simulators, we decided to observe NS-2 and GloMoSim more closely since OPNET is a commercial packet, while the others are open source.

Although we've studied GloMoSim and NS-2, in our research we decided to use NS-2 since it is more frequently used in worldwide network societies employs a larger number of ad hoc routing protocols and offers a complete support for the TCP/IP suite. In this paper we present our experiences gained with the use of NS-2 for simulations of wireless mobile ad hoc networks. We observed the flexibility of NS-2 simulator, by the means of adding new application layers, change of existing routing protocols, and incorporating various extensions created outside the NS-2 development team. We also explored the possibility of creation of various mobility and communication scenario generators.

This research represents is extensively used in our current research projects regarding the wireless mobile ad hoc networks.

2 Network simulators

Simulation is the execution of a system model in timely manner with the aim to obtain information about the system being investigated. For the results of the simulation to be meaningful, it is important that the model on which the simulator is based matches as closely as possible to reality. Because of our decision to use NS-2, we only give a short description for GloMoSim.

2.1 GloMoSim

GloMoSim is a scalable simulation environment for wireless and wired networks systems developed initially at UCLA Computing Laboratory [9]. It is designed using the parallel discrete-event simulation capability provided by a C-based parallel simulation language, Parsec [10]. GloMoSim currently supports protocols for purely wireless networks. It is build using a layered approach. Standard APIs are used between the different layers. This allows the rapid integration of models developed at different layers by users. In addition, GloMoSim provides various applications (CBR, ftp, telnet), transport protocols (TCP, UDP), routing protocols (AODV, ooding) and mobility schemes (random waypoint, random drunken). GloMoSim source and binary code can be downloaded only by academic institutions for research purposed. Commercial users must use QualNet, the commercial version of GloMoSim.

2.2 NS-2

The network simulator NS-2 is an object-oriented, discrete event-driven network simulator developed at UC Berkeley as part of the VINT project [8]. It is a very useful tool for conducting networks simulations involving local and wide area networks, but its functionality has grown during recent years to include wireless networks and ad-hoc networking as well.

The ns-2 network simulator has gained an enormous popularity among participants of the research community, mainly because of its simplicity and modularity. It allows *simulation scripts*, also called *simulation scenarios*, to be easily written in OTcl a script-like programming language. More complex functionality relies on C++ code that either comes with NS-2 or is supplied by the user. This flexibility is supposed to easily enhance the simulation environment as needed, although most common parts are already built-in, such as wired nodes, mobile nodes, links, queues, agents (protocols) and applications. Most of the network components can be configured in detail, and models for traffic patterns and errors can be applied to a simulation to increase its reality. There even exists an emulation feature, allowing the simulator to interact with a real network.

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. Simulations in NS-2 can be logged to trace files, which include detailed information about the received and sent packets in the simulation and allow post-run processing with some analysis tool. It is also possible to let ns-2 generate a special trace file that can be used by NAM (Network Animator), a visualization tool that is part of the NS-2 distribution. This allows simulations to be replayed on screen, which can be useful for complex simulations and education purpose.

3 NS-2 Observation

3.1 Output Trace Files Structure

In order to investigate the trace files that NS-2 generates as a result for each simulation, we have created and tested a number of different ad hoc network scenarios. Each of them had different input parameters such as node number variation, sent packets number variation, different routing protocols etc. The overall goal of our experiments was to measure the capability of the network simulator and to obtain sufficient data about the output trace file format and their meaning. By analyzing every single row from the generated trace files and taking into consideration the NS-2 documentation considering the trace files [7], we managed to clarify every element and get a clear view of every moment in the simulation. The different scenarios helped us to define the meaning of each field in the output trace file. Running number of diverse scenarios is crucial in order to enrich the information about the way the simulator reacts on different scenario and communication situations.

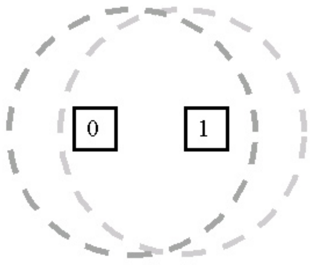


Figure 1: Ad hoc network simulation scenario of two reachable nodes

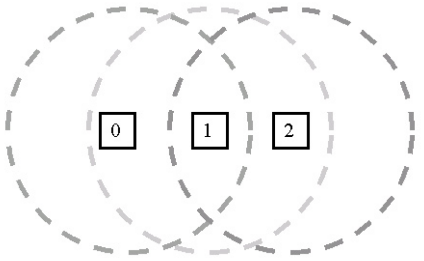


Figure 2: Simulation scenario of ad hoc network of three reachable nodes

Several simulation scenarios with two nodes on different positions were made. The next step was simulating scenario with three nodes. We made different scenarios with AODV and DSR routing protocols, different number of packets, different size of packets, different interval of generating the packet etc. Some of the scenarios are shown on Figure 1 and 2.

When each data packet is originated, an internal mechanism of the simulator (separate from the routing protocols) calculates the shortest path between the packet's sender and its destination. Each field in the trace file lines gives important information about the packet and it movement. We can monitor information about the event type, event time, source node, destination node, packet type, packet size, etc. An example output trace file row is shown on Figure 3.

```
s -t 10.936679000 -Hs 0 -Hd -2 -Ni 0 -Nx 250.00 -Ny 250.00 -Nz 0.00 -
Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -
It cbr -Il 32 -If 0 -Ii 0 -Iv 32 -Pn cbr -Pi 0 -Pf 0 -Po 0
```

Figure 3. One row from an example output trace file generated using NS-2 simulator for ad hoc networks simulations

The size of the output files is in accordance with the complexity of the scenario and its simulation time duration. The time of execution of the simulation is proportional to the size of the output trace file.

3.2 Ad Hoc Routing Protocols Implemented in NS-2

The routing protocol is a very important part of the ad hoc network layered architecture. The primary goal of the routing protocol is correct and efficient route establishment between pair of nodes so that messages can be delivered in a timely manner. NS-2 supports four ad hoc routing protocols: DSDV, DSR, TORA and AODV. The protocols are designed according to the IETF draft designs without their exact implementation. That is, some of the characteristics of the routing protocols in NS-2 are not matching to those from the IETF drafts. AODV as a reactive distance vector protocol, requests a route only when needed and does not require nodes to maintain routes to destinations that are not communicating. DSR is a reactive routing protocol that uses source routing to deliver data packets. In order to observe the behavior of NS-2 when using different types of routing protocols several different types of scenarios with different routing protocols were made. However, we collided with the problem of being unable to run some of the simulations when using DSDV and TORA because of errors in their implementation. In some of the cases the simulation ends with an error message, while for other scenarios the results are not corresponding to the expected behavior even for some trivial cases of node communication. Thus, we decided to use AODV and DSR as the routing protocols for all our future work. This choice was emphasized with the fact that these two protocols are also the most frequently used ones.

The scenarios for some of our simple test simulations are as follows: DSR or AODV routing protocol, two agents: UDP and NULL, constant bit rate - CBR source traffic, while the packets are being sent every 10 sec.

After detail analysis of the results a visualization of the packet movement between the layers was made (see Fig. 4 and Fig. 5).

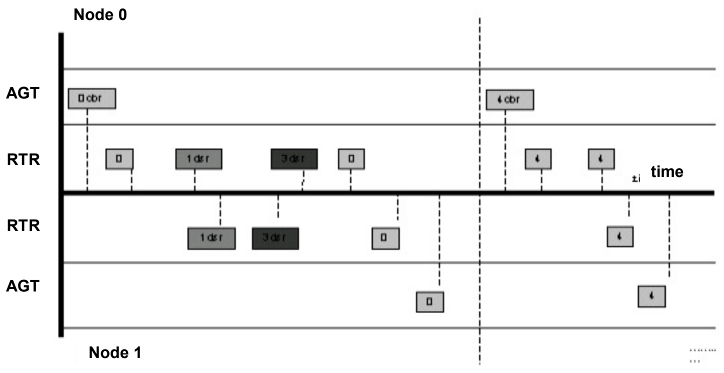


Figure 4: Packet movement in simulation scenario of two reachable nodes using DSR

3.3 New Application Layer Implementation

NS-2 has several different application levels that can be used while simulating ad hoc networks like CBR, Telnet, FTP and Ping. For the purposes of obtaining the ad hoc networks performances when the small world phenomena is taken into consideration [11], a new small world application layer was implemented since NS-2 does not include an application level protocol that is aware of any logical cluster division of a population of nodes, we created custom type of application level protocol that has this feature. By the means of this application level, each node in a given population knows which are its friends, or, more exactly, which are the nodes it can communicate with.

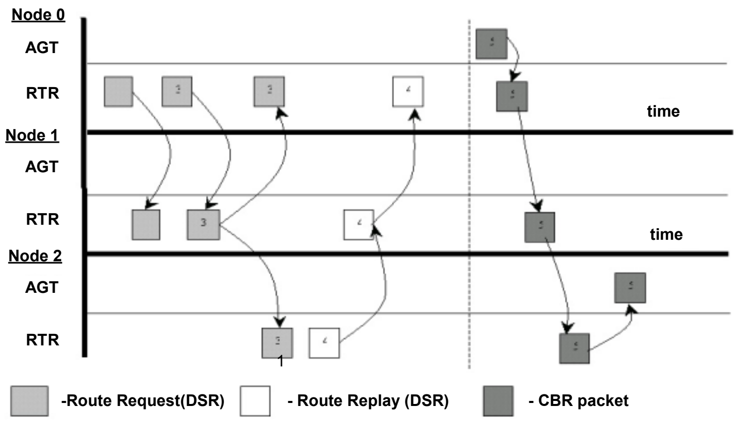


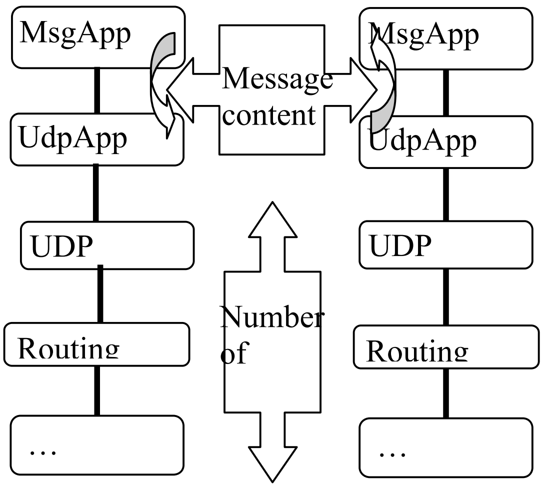
Figure 5: Packet movement in simulation scenario of three reachable nodes with DSR

The mobile nodes send messages to their friend nodes, which decide about their next action upon the message content. Trying to implement this behavior we came to understand that although there is an implementation wrapper command for sending the message content, it not passed to the lower level protocols. Therefore, they actually do not "travel" from the source to the destination, but are maintained at the source. The only thing that is really exchanged is the message size in bytes. In order to get the full message passed from the source, after the destination's application level receives the message size, it needs to explicitly connect to the source's application level to get access to the message content. All messages sent by a given source are maintained in linked list. Fig. 6 shows the network model that is used, and the flow of data between two nodes in the process of communication.

This experience has shown that implementing a new feature, like a new application layer in this case, is not a very simple, but still is a promising task. The new application layer was completely written in C, but several connections in C and OTcl needed to be made in order to "attach" the new layer to the simulator. The already implemented layers are of big assistance in these cases especially if one can be taken as the basis on which the newly designed feature will be upgraded.

3.4 Tools For Automatic Script Generation

In addition to NS-2, there are several independent tools for different types of tasks usually needed when using NS-2, like scenario and connection scripts generators and tools for post-simulation analysis of the generated trace files. Since the usual scenarios for simulations regarding ad hoc networks involve a bigger number of nodes scattered in a given rectangular area, there is a real need of tools that will be able to generate scenarios for node positioning and for the communications that will take place in the simulation.



The script generators that come with the NS-2 distribution can be used to create node positioning scenario scripts based on the area size, number of nodes, simulation time, node speed and node pause time. This scenario generator uses these input values to randomly scatter the nodes using uniform distribution and than implements the Random Walk Mobility Model for the node movements during the simulation. Other types of Mobility Models can be found as a contribution code for NS-2 and need some adjustments in order to be used. The connection scripts generator, on the other hand, is even more restricted than the one previously described. The user can only choose between random TCP and CBR traffic while marking the maximum number of connections at a given moment. This implies that the creation of more sophisticated and demanding simulation scripts is left to the user. This is also the case when it comes to extracting needed information out of the obtained output trace files, since NS-2 does not provide the solution of this problem except for the Network Animator – NAM that can be used for visualization.

3.5 Source Code Changes and External Code Adaptations

The NS-2 open source nature allows the user to change the NS-2 features in order to modify the way NS-2 deals with the network simulation. This is a very useful characteristic that allows running various types of simulations while varying some inner NS-2 characteristic that otherwise would not be a changeable parameter or running simulations with changed layers features and mechanisms of work. Of course the NS-2 open source nature can be used only when the user is able to find his way around the C and OTcl code. In our research we made changes on some features of AODV and DSR routing protocols in order to improve their performances.

One can found lots of contribution code made for some version of NS-2 which is made by other NS-2 users and has very interesting implementations. Trying to add an external code to NS-2, however, has proven to be a more exhausting work that imagined. Although the code that needs to be added is made especially for NS-2 the version variation may be fatal to the users merging attempts. During the recent period NS-2 has suffered great changes and, therefore the adaptation of some old code to NS-2 new versions has become quite a work, making the adaptation very hard to manage. Our experience was gained while trying to adapt the Clustering Based Routing Protocol - CBRP and Zone Routing Protocol – ZRP made for an earlier NS-2 versions. The progression from the early NS-2 versions to the most recent one has made severe changes in the NS-2 organization and implementation. This resulted in a complete go-through the old codes in order to make the necessary adjustments. The lack of necessary documentation for these problems has made this task even more difficult.

4 Conclusion

The area of ad hoc networking has been receiving increasing attention among researchers in recent years. With the increased attention the need for good simulators for ad hoc networks rises. Taking into consideration several of the most popular network simulators, we decided to use NS-2 it is the most widely used and employs a number of ad hoc networks features like several ad hoc routing protocols and wireless layer support. In this paper we presented our experienced obtained with the use of NS-2 for creating and running various types of ad hoc network simulations. We explored the NS-2 features by taking advantage of its open source nature. However, some disadvantages of NS-2 stem directly from its open source nature. The provided documentation is limited and out of date with the current release of the simulator, therefore being of very little help to the user. Facing the problem of poor documentation for ad hoc networks, we made a number of simulations that allowed us to create a knowledge database that is extensively used in our current research projects regarding the wireless mobile ad hoc networks.

5 References

1. Charles E. Perkins, *Ad Hoc Networking*, Addison – Wesley 2001
2. R. Ramanathan and J. Redi, “A Brief Overview of Ad Hoc Networks: Challenges and Directions”, *IEEE Communications Magazine*, May 2002
3. David Cavin, Yoav Sasson, André Schiper, “On the Accuracy of MANET Simulators”, *POMC’02*, October 2002
4. *The network simulator – ns-2*, http://www.isi.edu/nsnam/ns
5. GloMoSim (ver. 1.2), http://pcl.cs.ucla.edu/projects/glomosim
6. *OPNET Modeller*, http://www.opnet.com
7. *ns-2 Trace Formats*, http://www.eecs.wsu.edu/~rgriswol/NS2/ns2-trace-formats.html