Global Mobile Information System Simulator in Fedora Linux

Ayyaswamy Kathirvel [a] , Rengaramanujam Srinivasan[b]

a Assistant Professor, *Faculty of Computer Science and Engineering*
b Professor, Faculty of Computer Science and Engineering
EmailID: a kathir@crescentcollege.org, *b rsvaan@crescentcollege.org*
B.S.Abdur Rahman University, Chennai – 600 048, Tamilnadu, India.

Abstract:
Global Mobile Information System Simulator is a popular network simulation tool, which is frequently used in the study of the behavior of large-scale hybrid networks that include wireless, wired, and satellite based communications are becoming common in both in military and commercial situations. It is freely available without fee for education, or research, or to non-profit agencies. It is simple to install and use. It is available for various Linux flavors files include freebsd-3.3, aix, irix, redhat-6.0, redhat-7.2 and Solaris. This tool is not supported to fedora core Linux. This paper will help you get started.

Keywords: GloMoSim, Visualization Tool, GnuPlot and Parsec.

1. INTRODUCTION

Global Mobile Information System Simulator (GloMoSim) simulates networks with up to thousand nodes linked by a heterogeneous communications capability that includes multicast, asymmetric communications using direct satellite broadcasts, multi-hop wireless communications using ad-hoc networking, and traditional Internet protocols. Developers use these simulators to model the wired or wireless network design process [1][2]. It is being designed using the parallel discrete-event simulation capability provided by Parsec. This makes it possible to evaluate various design alterations and configurations before even deploying the actual devices and components. Based on the outcome of the validation process, simulations can once again be attempted for optimization of the hardware performance.

Most network systems are currently build using a layered approach that is similar to the OSI seven layer network architecture. The plan is to build GloMoSim using a similar layered approach [4]. Standard APIs will be used between the different simulation layers. This will allow the rapid integration of models developed at different layers by different people [13][14]. It usually made available on a standalone machine. The goal is to build a library of parallelized models that can be used for the evaluation of variety of wireless network protocols [5][7]. The proposed protocols stack will include models for the channel, radio, MAC, network, transport, and higher layers.

The simple approach to designing a network simulation would be to initialize each network node in the simulation as a Parsec entity [6]. We can view different entity initializations as being separate logical processes in the system. Hence each entity initialization requires its own stack space in the runtime. In GloMoSim, we are trying to build a simulation that will scale to thousands of nodes [4]. If we have to instantiate an entity for each node in the runtime, the memory requirements would increase dramatically [15]. The performance of the system would also degrade rapidly. Since there are so many entities in the simulation, the runtime would need to constantly context switch among the different entities in the system. This will cause significant degradation in the performance of the simulation [8][9]. Hence initializing each node as a separate entity will inherently limit the scalability and performance of the simulation [16][17].

To circumvent these problems network gridding was introduced into the simulation [18]. With network gridding, a single entity can simulate several network nodes in the system. A separate data structure

representing the complete state of each node is maintained within the entity. Similarly we need to maintain the right level of abstraction. When the simulation code of a particular node is being executed it should not have access to the data structures of the other nodes in the simulation. The network gridding technique means that we can increase the number of nodes in the system while maintaining the same number of entities in the simulation [19]. In fact, the only requirement is that we need only as many entities as the number of processors on which the simulation is being run. Hence if we are running a sequential simulation we need to initialize only one entity in the system. We also don't meet the memory or context switching problems that limit the simulation [20]. The rest of the paper is organized as follows. In the following section, we give the software required to install GloMoSim. Section 3 describes the Visualization Tool. Section 4 describes the simulation Result analysis. We draw our conclusions in Section 5.

2. SOFTWARE REQUIRED TO INSTALL GLOMOSIM

To install GloMoSim Software we required following software's include

1.Linux of Java 1.3 or higher versions.
2.Parsec compiler
3.GloMoSim software

Usually GloMoSim software is come along with parsec compiler if so no need to download parsec compiler separately. Otherwise download parsec compiler separately.

GloMoSim tool and Parsec compiler can be downloaded free of cost from the URL [http://pcl.cs.ucla.edu/projects/glomosim/] also for online help refer to URL [http://pcl.cs.ucla.edu/projects/parsec/] and URL [http://pcl.cs.ucla.edu/projects/glomosim]. The tool has long been available for various Linux flavors include freebsd-3.3, aix, irix, redhat-6.0, redhat-7.2 and Solaris. Though GloMoSim are available for the Windows platform, it is recommended you install the tool on Linux. Similarly java [10] compiler for Linux can be downloaded free of cost from the URL [http://www.java.com/en/download/manual.jsp].

2.1 Installing GloMoSim

Before installing GloMoSim you have to install java for Linux. Simply double click jdk-1_5_0_07-linux-i586-rpm.bin it shows at fig1 the following screen, press Run in Terminal button. It is automatically installed at the directory /usr/java/jdk1.5.0_07

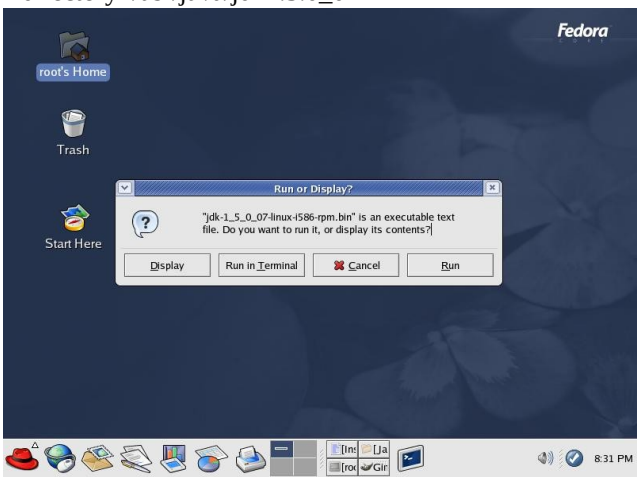

Fig 1: Installation of jdk1.5 in Linux

After downloading the tar.gz file, simply issue the Linux commands listed below at the command prompt, to install the tool.

#tar xvfz glomosim-2.03.tar.gz
// to uncompress the glomosim software archive

This will create a new directory under the root directory named glomosim-2.03. Inside the glomosim-2.03 directories, which consist of two subdirectories, can be found namely glomosim and parsec. Inside the parsec directory various Linux flavors files include freebsd-3.3, aix, irix, redhat-6.0, redhat-7.2 and Solaris. Note that downloaded software is not supported to fedora. If you are run this tool in fedora, copy all files inside the redhat-7.2 directory, paste it in /parsec directory. You can find the .makefile(make) executable file at the following directory /glomosim-2.03/glomosim/main.

#cd /glomosim-2.03/glomosim/main

#make

vi .bash_profile

// to open the .bash_profile to customize glomosim software

2.2 Customizing GloMoSim

As such, the path of glomosim could be anywhere on the file system. However, it is recommended you install glomosim on the file system of a user other than root. In order to allow glomosim to run from the user's file system, you may need to make a few changes to the .bash_profile file under the home directory of the user. This will also allow you to use glomosim without having to log in as the all-powered root user. The following changes then should be made to the file .bash_profile.

User specific environment and startup programs

PATH=$PATH:$HOME/bin:/usr/java/jdk1.5.0_07/bin:/glomosim-2.03/glomosim/main :/glomosim-2.03/glomosim/include:/glomosim-2.03/glomosim/bin:/glomosim-2.03/parsec/bin:/glomosim-2.03/parsec/include

PCC_DIRECTORY=/glomosim-2.03/parsec
export PATH PCC_DIRECTORY

It may be noted that the installation process in its final response will suggest these changes.

2.3 Simulate a network using glomosim

After successfully installing GloMoSim, a simulation can be started by executing the following command in the BIN subdirectory.

./glomosim inputfile > bell.trace

The inputfile contains the configuration parameters for the simulation (an example of such file is CONFIG.IN). A file called GLOMO.STAT is produced at the end of the simulation and contains all the statistics generated.

3. THE VISUALIZATION TOOL

GloMoSim has a visualization tool that is platform independent because it is coded in java [3]. To initialize the visualization tool, we must execute from the java_gui directory the following: java GlomoMain. This tool allows to debug and verify models and scenarios; stop, resume and stop execution; show packet transmissions, show mobility groups in different colors and show statistics at fig2, fig3 and fig4.

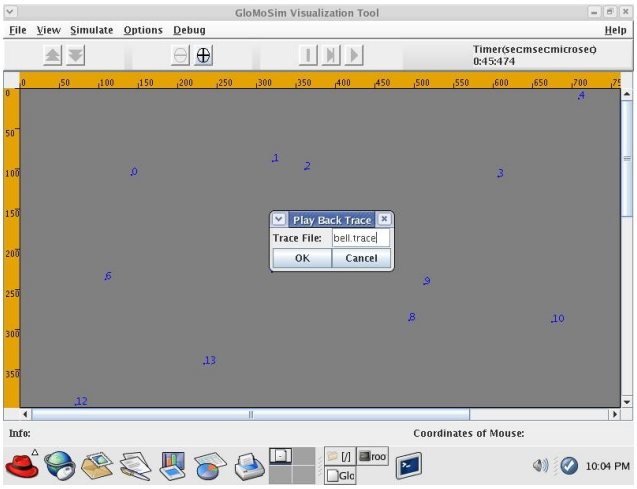

Fig 2: Visualization Tool to open bell.trace file

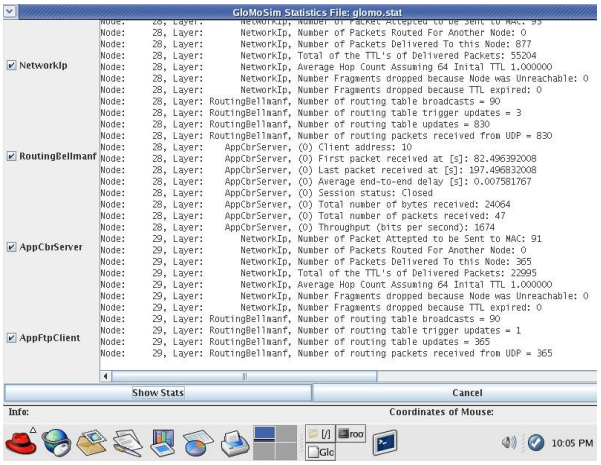

Fig 3: Visualization Tool to run bell.trace file

The radio layer is displayed in the visualization tool as follows: when a node transmits a packer, a yellow link is drawn from this node to all nodes within its power range. As each node receives the packet, the link is erased and a green line is drawn for successful reception and a red line is drawn for unsuccessful reception. No distinction is made between different packet types (ie. Control packets vs regular packets etc). Note that bell.trace file which is

available at ./bin directory copied into ./java_gui directory.

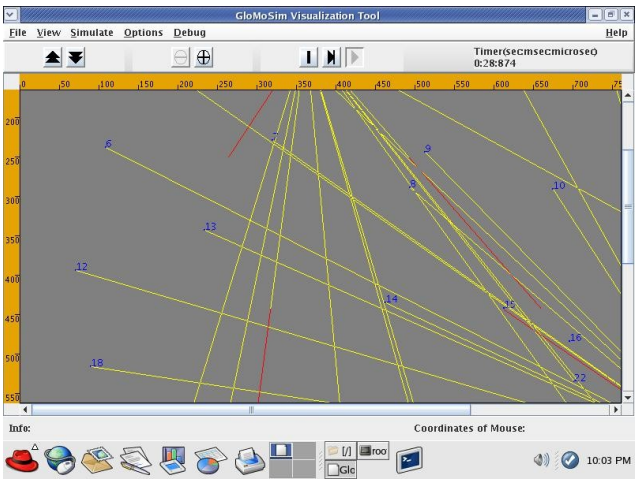

Fig 4: GloMoSim Statistics File to open glomo.stat file

4. RESULTS ANALYSIS

The output data of the network is generated and stored in glomo.stat file. In these cases we use other tools to do an analysis of the glomo.stat file. For example, the following shell program and awk programs is used to display packet data ratio, percentage of loss rate and packets sent.

Shell program:
File Name : result1.sh

```
Cat $1 | grep App | grep Total | grep packets | awk –f analysis.awk
```

AWK program:
File Name: analysis.awk

```
BEGIN{
      sumcountsent = 0;
      sumcountrcv  = 0;
      }
      {
      if($10 =="sent:") sumcountsent+=$11;
        else if ($10 == "received:") sumcountrcv+=$11;
      }
  END{
        printf("Loss Packet Percentage= %f \% \n",100-((sumcountrcv*100)/sumcountsent));
      printf("Packet Delivery Ratio  = %f \n\n",sumcountrcv/sumcountsent);
      printf("Packet Received = %d \n", sumcountrcv);
      printf("Packet Sent  = %d \n\n", sumcountsent);
      }
```

An example of the execution of these programs the result as follows

```
#sh result1.sh glomo.stat

Loss Packet Percentage = 36.891678 %
Packet Delivery Ratio   =  0.678990
Packets Received        =  15233
Packets Sent                = 24300
```

To obtain the Average Delay and the Control Packets generated by the AODV routing protocol, the following programs can be used:

Shell program:
File Name : result2.sh

```
cat $1 | grep AppCbrServer | grep end-to-end | grep delay | awk -f delay.awk
```

AWK program:
File Name: delay.awk

```
BEGIN{
        sumdelay=0;
        countdelay=0;
    }
    {
     sumdelay +=$10;
     countdelay++;
    }
END{
            printf("Average Delay = %f \n", sumdelay/countdelay);}
```

An example of the execution of these programs the result as follows

```
#sh result2.sh glomo.stat
Average Delay = 12.191678 %
```

We can even obtain some interesting graphs from simulations. Lets suppose we want to obtain a graph of the variance of packets delivered to final destination, with networks of different size (30,50,70,100 and 200 nodes) and nodes move at different speeds (1,5,10 amd 20 m/s). We can run several simulations (each with a specified node number and speed of the mobility model). To shorten the simulation time, the tests can be executed at the same time, thus obtaining the following table.

speed	30 nodes	50 nodes	70 nodes	100 nodes	200 nodes
1	12884	34890	56899	578909	67989
5	15927	33098	43898	37989	67700
10	14989	22890	34900	34900	49090
20	12322	20323	29000	23798	29009

If the data of this table is used in a file (e.g data.txt), then we may use graphic tools such as GnuPlot [11] to obtain an appropriate graph. In the case of GnuPlot [12], we can execute the following commands to obtain the graph represented at fig5 and contained in file graph.eps.

```
set terminal postscript eps
set size 1/1., 1/1.
set title " Performance"
set output "graph.eps"
set xlabel "Speed (m/s)"
set ylabel "Packets"
plot [0:20][8000:75000] "data.txt" using 1:2 title '30 nodes' with linespoints, \
                        "data.txt" using 1:3 title '50 nodes' with linespoints, \
                        "data.txt" using 1:4 title '70 nodes' with linespoints, \
                       "mobil.txt" using 1:5 title '100 nodes' with linespoints, \
                        "data.txt" using 1:6 title '200 nodes' with linespoints
```

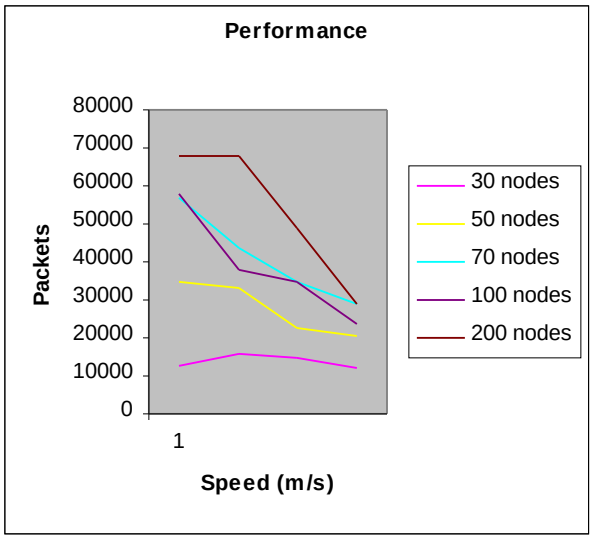

Fig 5: Packet delivery ration using Gnu Plot

GloMoSim is a popular simulation tool that is freely available for education, or research, or to non-profit agencies, which means you can enhance it to suit your own requirements.

5. CONCLUSION

Detailed, high fidelity models of large networks represent a significant challenge for the networking community. This paper presented a simulation library called GloMoSim whose goal is to support accurate performance prediction of large-scale network models using parallel execution on a diverse set of parallel computers. The library has already been used to simulate networks with thousands of wireless nodes and provides a rich set of models for both existing and novel protocols at multiple layers of the protocol stack.

6. ACKNOWLEDGEMENTS

The authors would like to express their thanks to Dr. K. M. Mehata, the HOD and Prof.V.M.Periasamy, the Registrar of B.S.A. University, Chennai for the environment provided.

REFERENCE

1. GloMoSim: Global Mobile Information Systems Simulation Library. http://pcl.cs.ucla.edu/projects/glomosim/
2. Mario Gerla Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Rajive Bagrodia. GloMoSim: A Scalable Network Simulation Environment. Technical Report 990027, University of California, 13, 1999.
3. Addison Lee- Kaixin Xu. GloMoSim Java Visualization Tool. Documentation version 1.1, Software Distribution.
4. Rajive Bagrodia. README file – GloMoSim software. University of California, Los Angeles 90095-1596.
5. Theodore S. Rappaport. Wireless Communications: Principles and Practice. Prentice Hall, New Jersey, 1999.
6. R. Bagrodia, R. Meyer et al. PARSEC: A Parallel Simulation Environment for Complex Systems. IEEE Computer, 98.
7. R. Bagrodia, Y. A. Chen et al. Parallel Simulation of a High-speed Wormhole Routing Network. Proceeding of 10th Workshop on Parallel and Distributed Simulations, PADS 96.
8. S. Bhatt, R. Fujimoto, A. Ogieski and K. Permalla. Parallel Simulation Techniques for Large Scale Networks. IEEE Communication Magazine, 98, pp. 42-47.
9. Zygmunt J. Haas et al. Wireless Ad Hoc Networks. In Encyclopedia of Telecommunications, 2002.
10. Java software for Linux free download: http://www.java.com/en/download/manual.jsp
11. Gnuplot. A Brief Manual and Tutorial. University of Duke. www.duke.edu/~hpgavin/**gnuplot**.html
12. Gnuplot. Documentation version 4.0, Software Distribution. www.**gnuplot**.info/
13. Mobile Ad Hoc Networks (MANET) Working Group, http://www.ietf.org/html.characters/manet-charter.html,2004
14. Karoly Farkas, Dirk Budke, Bernhard Plattner, Oliver Wellnitz and Lars Wolf. QoS Extensions to Mobile Ad hoc Routing Supporting Real-Time Applications. ACM, 2004.
15. C.E.Perkins, E.M. Belding-Royer, and I.D.Chakeres. Ad hoc on demand distance vector (aodv) routing. IETF Internet draft, oct. 2003.
16. D.B. Johnson, D.A.Maltz, and Y.c.Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr). Internet draft, draft-ietf-manet-dsr-09.txt,apr 2003.
17. Satyabrata chakrabarti and Amitabh mishra. Quality of service in Mobile Ad hoc Networks. CRC press LLC.2003
18. Kui Wu and Jamelle Harms. On-Demand Multipath Routing for Mobile Ad hoc Networks. Proceeding of EPMCC\ACM. 2001
19. Yih-Chun Hu and David B.Johnson. Implicit Source Routes for On-Demand Ad hoc Network Routing. ACM 2001.
20. IEEE 802.11: part 11: Wireless LAN Medium Access control (MAC) and Physical Layer (PHY) specification, Aug.1999.

Author Biographies:

A.Kathirvel - born in 1976 in Erode, Tamilnadu, India, received his B.E. degree from the University of Madras, Chennai, in 1998 and M.E. degree from the same University in 2002. He is currently with B.S.Abdur Raahman University, in the Department of computer science and Engineering and pursing Ph.D. degree with the Anna University, Chennai, India. He is a member of the ISTE. His research interests are protocol development for wireless ad hoc networks, security in ad hoc networks.

Rengaramanujam Srinivasan -- born in 1940 in Alwartirunagari, Tamilnadu, India, received B.E. degree from the University of Madras, Chennai, India in 1962, M.E. degree from the Indian Institute of Science, Bangalore, India in 1964 and Ph.D. degree from the Indian Institute of Technology, Kharagpur, India in 1971. He is a member of the ISTE and a Fellow of Institution of Engineers, India. He has over 40 years of experience in teaching and research. He is presently working as a Professor of Computer Science and Engineering at B.S.Abdur Raahman University, Chennai, India and is supervising doctoral projects in the areas of data mining, wireless networks, Grid Computing, Information Retrieval and Software Engineering.