Idris Skloul Ibrahim*, Peter J.B. King and Hans-Wolfgang Loidl

NsGTFA: A GUI Tool to Easily Measure Network Performance through the Ns2 Trace File

Abstract: Ns2 is an open-source communications network simulator primarily used in research and teaching. Ns2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. Although Ns2 is a widely used powerful simulator, it lacks a way to measure networks that are used to assess reliability and performance metrics (e.g., the number of packets transferred from source to destination, delay in packets, packet loss, etc.) and it does not analyse the trace files it produces. The data obtained from the simulations are not straightforward to analyse. Ns2 is still unable to provide any data analysis statistics or graphics as requested. Moreover, the analysis of the Ns2 trace file using any software scripts requires further steps by a developer to do data processing and then produce graphical outputs. Lack of standardisation of tools means that results from different users may not be strictly comparable. There are alternative tools; however, most of them are not standalone applications, requiring some additional libraries. Also, they lack a user-friendly interface. This article presents the architecture and development considerations for the NsGTFA (Ns2 GUI Trace File Analyser) tool, which intends to simplify the management and enable the statistical analysis of trace files generated during network simulations. NsGTFA runs under Windows and has a friendly graphical user interface. This tool is a very fast standalone application implemented in VC++, taking as input an Ns2 trace file. It can output two-dimensional (2D) and 3D graphs (points, lines, and bar charts) or data sets, whatever the trace file format (Tagged, Old, or New). It is also possible to specify the output of standard network performance metrics. NsGTFA satisfies most user needs. There is no complex installation process, and no external libraries are needed.

Keywords: VC++, GUI trace file analyser, trace file format, network performance metrics.

DOI 10.1515/jisys-2014-0153
Received October 17, 2014; previously published online January 15, 2015.

1 Introduction

Ns2 is a discrete event simulator [5] written in C++ and visualised with OTcl [9] based on an object-oriented concept and design, which was developed as a part of the VINT project [5, 14]. It is a DARPA-funded research project whose aim is to provide a reliable network simulator for scale and protocol interaction study based on the present and future network and protocols. It supports simulation of IP, TCP, UDP, routing, and multicast protocols; quality of service mechanisms; and more. Although Ns2 is a powerful and widely used simulation tool [2, 4, 5], the analysis of the data obtained from the simulations is not straightforward. Users can visualise the network layout and packet level operations using the Network Animator (a Tcl/TK-based animation tool). Ns2 does not provide any data analysis (performance metrics statistics or graphics). Moreover, the parsing of the Ns2 trace file using Java, C, or AWK scripts developed by users requires a further step to do data processing and graphical organisation. Users need a standalone application that is able to easily translate the Ns2

***Corresponding author: Idris Skloul Ibrahim,** School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, Scotland, UK, e-mail: i.s.ibrahim@hw.ac.uk
Peter J.B. King and Hans-Wolfgang Loidl: School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK

trace files into graphics and to give a summary of what occurred during simulations; this has motivated the development of a new data analysis tool reported in this article (NsGTFA – Ns2 GUI Trace File Analyser).

NsGTFA is an application capable of analysing Ns2 normal wireless trace files along with Tagged, Old, and New formats. This tool is fast enough to read millions of events in seconds, and capable of analysing the trace files (performance metrics) and making graphics and statistics about the behaviour of the flows in simulation. The main characteristics and functionalities are described in the next sections, along with some examples.

2 Related Work

The Ns2 log or trace files gather the required information that can be visualised, such as in the NAM animator [4], or could be used in a network and protocol performance study, for example, the amount of packets transferred, delay, and packet loss. These event log files contain packets being sent, received, or only dropped, the so-called packet traces. In fact, the trace files are just a text format; in other words, they are human readable, where we can access them using some form of offline software. To ease the process of extracting data for performance studies, many Ns2 trace analysers have been proposed [1, 8, 11, 12].

2.1 Jtarana

Jtarana is developed in 2011; it requires Java and MYSQL. It can be used to analyse Ns2 trace files, generates many types of graph and statistics, and is easier to use than other tools such as the Gnu plot. Unfortunately, Jtarana is very slow [8]: reading a 2 MB file in 10 min and a 20 MB file in 100 min. Even after successfully reading the file, when it generates graphs it becomes much slower; however, it does show correct readings and excellent graphs.

Ns2 Visual Trace Analyser is a visual GUI for Ns2 trace and scenario files, developed in 2009 [11]. It allows users to trace graphics, filter packets, visualise node positions, calculate node and traffic statistics, and so on.

Ns2 Visual Trace Analyser requires loading the TCL file (full details are found in References [5, 6]), so that the application can build the topology and get some extra parameters and the trace file, with all packet information. This tool needs to upload the scenario (TCL) and trace files (tr), respectively, to start analysis and visualisation; this is not helpful because sometimes the user does not need to know the scenario topology specifically in huge and random scenario files. Unfortunately, the user will be unable to analyse the trace file unless they load the scenario file first. Also, perhaps the user loads unrelated scenario and trace files by mistake.

2.2 TRAFIL

TRAFIL is a Java tool [1]. It focuses on Ns2 trace files but can be extended to handle a variety of simulation trace file formats. The main benefit of TRAFIL apart from the production of ready-to-use metrics and charts is the reduced time it consumes for the whole trace file analysis procedure; first, by minimising the time to open and use a trace file and, second, by enabling the user to store each trace file in a database for easy access and future use.

3 Ns2 Trace File Logs

The network simulator Ns2 generates event logs called trace or log files [6], which can be analysed after a simulation of each scenario (offline). The log files gather information that can be used in a performance study,

and record and provide information for events of packets being sent, received, or dropped. This is called the packet trace. These log files collect all the necessary information needed for the analysis and evaluation of a protocol's performance. Ns2 provides three different formats of log files described in the following sections.

In analysing network performance after network simulation, implementing or testing any routing protocol such as DSDV, AODV, DSR, or TORA [3, 7, 10, 13] on Ns2, you need a tool for processing Ns2 simulator output (trace file) to extract the required statistics. Ns2 offers three different formats of the trace file: Old, New, and Tagged trace formats (full details are found in References [5, 6]). The following are samples of the new format that has been generated by Ns2 simulations.

3.1 Normal Format (Old)

This is the most commonly used format because of its simplicity, its fields are grouped to provide different information from the packet's fields, and it is easy to read.

s 74.406840052 _6_ AGT --- 1755 ack 40 [0 0 0 0] ------- [6:0 1:0 32 0] [829 0] 0 0

r 74.406840052 _6_ RTR --- 1755 ack 40 [0 0 0 0] ------- [6:0 1:0 32 0] [829 0] 0 0

Event type: The first field is encoded as one small letter and describes the node's event type, which can be one of four characters: s: send, r: receive, d: drop, or f: forward.

3.2 New Format

This offers more information than the previous one on each event during the simulation time. Therefore, the length of its lines means it is not as easy to read as the Old trace format, which contains many fields and tag-value pairs.

f -t 53.431526201 -Hs 18 -Hd 2 -Ni 18 -Nx 260.19 -Ny 70.21 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 13a -Md 12 -Ms 1 -Mt 800 -Is 1.0 -Id 2.0 -It cbr -Il 532 -If 0 -Ii 641 -Iv 31 -Pn cbr -Pi 203 -Pf 1 -Po 2

d -t 57.431526201 -Hs 13 -Hd 3 -Ni 19 -Nx 160.17 -Ny 90.21 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 14d -Md 12 -Ms 1 -Mt 800 -Is 1.0 -Id 2.0 -It cbr -Il 532 -If 0 -Ii 641 -Iv 31 -Pn cbr -Pi 203 -Pf 1 -Po 2

General tag: The second field is the event's time, based on the simulation time, starting with "-t," which stands for time or global setting, -t: time.

Node property tags: These fields denote node properties like node-id and the level at which tracing is being done like agent, router, or MAC.

Packet information (at IP level): These field tags start with a leading "-I." The tags and explanations are listed as follows:

-Is: Source address; port number
-Id: Destination address port number
-It: Packet type (message, Beacon, RSUP)
-Il: Packet size
-If: Flow id
-Ii: Unique id
-Iv: TTL value

3.3 Tagged

This trace format is the latest to be added to Ns2. It is also contains tags and values similar to the new trace format. Tagged trace format analysis is more difficult than in the previous two formats because the tag names must be determined by each object to be traced. Therefore, tag clashes are expected and likely to occur.

r 2.00094040138649 -s 5 -d -1 -p AODV -k RTR -i 0 -N:loc {590.00 520.00 0.00} -N:en -1.000000 -M:dur 0 -M:s ffffffff -M:d 6 -M:t 800 -IP:s 6 -IP:sp 255 -IP:d -1 -IP:dp 255 -p AODV -e 48 -c 0 -i 0 -IP:ttl 30 -aodv:t 2 -aodv:h 1 -aodv:b 1 -aodv:d 9 -aodv:ds 0 -aodv:s 6 -aodv:ss 4 -aodv:c REQUEST

4 Performance Metrics

Processing Ns2 simulator output to get user-requested statistics for any network scenario files, NsGTFA reads the generated trace file and calculates the following primary performance metrics to measure network reliability and performance.

Packet delivery fraction (PDF): The ratio of the total received packets by constant bit rate (CBR is an encoding method that keeps the bit rate the same) sink at destination over the total sent packets by the constant bit rate source (CBR, "application layer").

$$PDF = \sum \textit{received data packets} / \sum \textit{sent data packets}$$

Throughput: The rate at which a network sends and receives data. It is a good channel capacity of network connections and rated in terms bits per second (bit/s). In other words, it is the amount of data (bits) transferred from the destination node to the source node during a specified amount of time (s).

Normalised routing load (NRL): The number of routing packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission.

$$NRL = \textit{routing packets} / \textit{data packet delivered}.$$

End-to-end delay: The average time taken by a data packet to arrive at the destination; the delay that could be caused by buffering during route discovery, queuing delays at interface queues, retransmission delays at the MAC address, and propagation and transfer times.

$$EED = \sum (\textit{arrive time} - \textit{send time}) / \sum (\textit{number of connections}).$$

Routing overheads: The sum of all the routing control packets sent during the simulation time. For all the forwarded packets over multiple hops, each packet transmitted over multihops counts as one transmission.

$$\textit{Routing overheads } (Oh) = \sum \textit{routing control packets sent}.$$

Packet loss: The number of packets that fail to reach their destination in a timely manner. Most commonly, packets get dropped before the destination can be reached.

$$\textit{Packet loss} = \textit{sent data packets} - \textit{received data packets}.$$

Jitter: The fluctuation of end-to-end delay from one packet to the next packet of connection flow.

5 NsGTFA Structure

In this section, we present NsGTFA's architecture along with modules that consist it. NsGTFA is the main module and it consists of three layers: the work layer, presentation layer, and data storage layer, as shown in Figure 1. The work layer is the highest and main layer. It consists of four modules: the file handler, metrics,

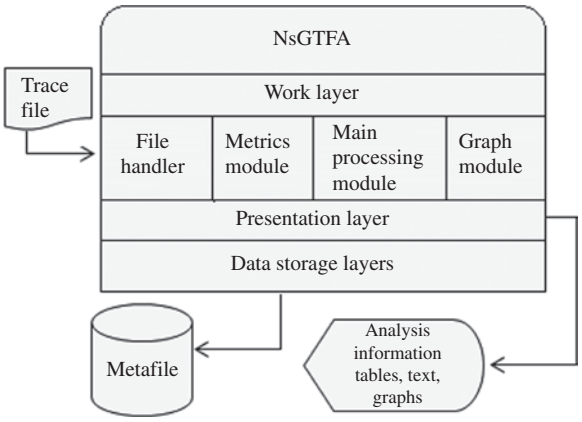

Figure 1. NsGTFA Structure.

main processing, and graph modules. This layer loads and processes the trace file (only Old, New, and Tagged format are loadable) and then passes the output information to the middle "presentation" layer. In the presentation layer, the output information is displayed either in table, report, or graph format as requested. Also, the output information can be stored in a metafile for easy access and future use in the lowest layer, "data storage."

6 Trace File Journey from Ns2 to NsGTFA

In the absence of formal mathematical proof of the correctness of the wireless network or routing protocol, we need an extensive set of tests (verification) to confirm that the wireless network meets the intended specifications and that it is fully functional and works in a wide range of environments. It is easy sometimes to convince yourself that it works, only to find that it fails in the random radio environment. One of the most common methods is functionality testing using log files (Trace, Nam) that are generated by the network simulator tool Ns2.

The trace file journey starts from mobile scenarios and traffic pattern generation as shown in Figure 2. Generating mobile scenarios and traffic pattern are very simple in Ns2 using the mobility scenario and a traffic pattern generator (full details are found in Reference [6]). Then, you need a TCL script file to read both the mobility scenario and the traffic pattern files to generate Nam and Trace log files [5, 6].

To visualise the simulation results and real-world packet traces, the Nam trace file is used by the visualisation tool (NAM). NAM is a TCL/TK-based animator used for replying events during simulation time. If the events happen intensively or the simulation time is long, the NAM trace file can be huge. Full details are found in References [5, 6].

To measure the network performance and to produce performance graphs, the Trace log file is used by the GUI trace file analyser tool (NsGTFA), as shown in Figure 2.

7 NsGTFA Use and Layout

The layout of NsGTFA is fully developed using VC++ (Microsoft Visual Studio 2010). It is designed in such a way that reports, graph, tables, and the main layout is in its own window as shown in Figure 3. The tool's layout consists of a main menu and two boards for data view (analysis results board and performance charts board).

Loading the trace file is the first stage in trace file analysis. The user can search in their file system (computer drivers) to select and load the trace file by clicking on the "Load Trace File" button.

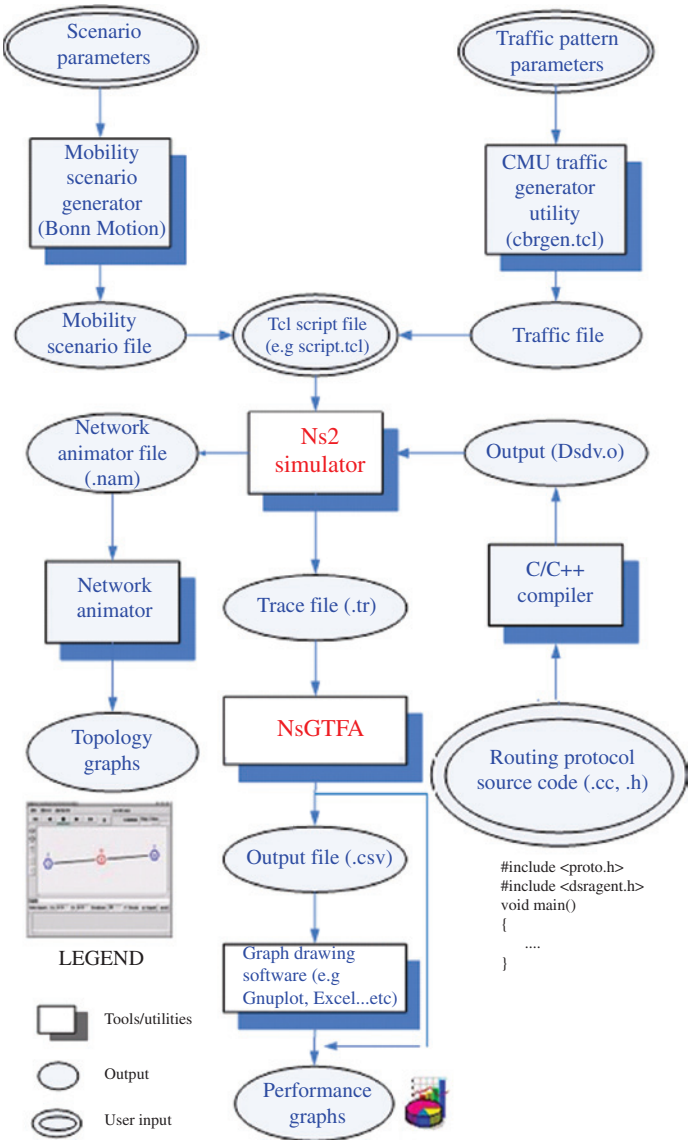

Figure 2. Trace File Flow Diagram in Ns2 and NsGTFA.

The five metrics mentioned previously are selected (ticked) by default, as shown in Figure 3. This means the selected trace file will be measured on the basis of all that five metrics. Also, the user has choices (options) to select/deselect or to clear all performance metrics.

The last process window shows the user the last process they performed – "File select," "Start analysis," "Trace file list," "Performance graphs," or "Results save." Additionally, if there is an error during the trace file read or results save, the "Error in" window indicates that.

The trace file can be very large. Thus, processing the trace file line by line can be an exhaustive task. Therefore, the process is multithreaded, buffered, and optimised to use the most computer processing power possible. In the second stage, "Analysis," a progress bar for informative window appears to show the current status of the trace file processing, total number of lines processed, and process start and end time, as shown in Figure 4.

At the end of the second stage, "Trace file analysis," NsGTFA displays a results report of ticked (selected) performance metrics for that trace file immediately on the "Results report board" along with some information about the selected trace file as shown in Figure 5.

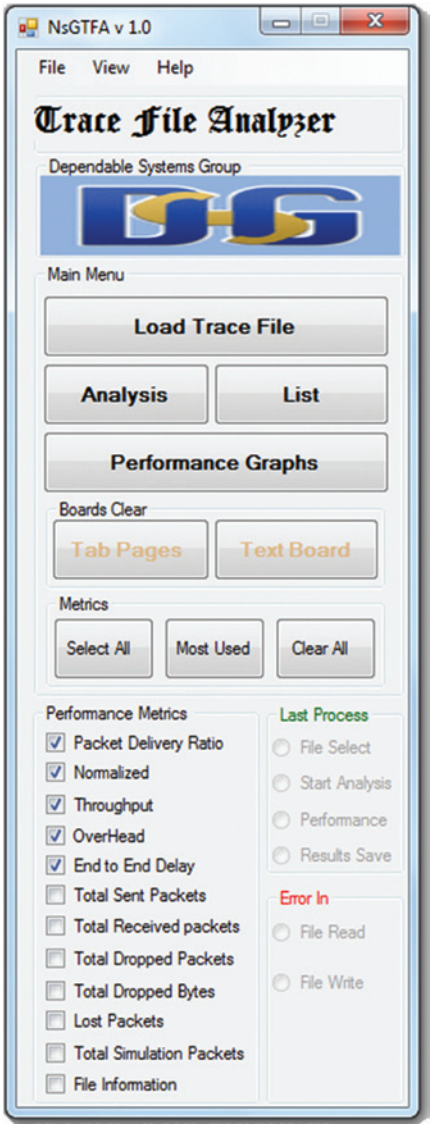

Figure 3. NsGTFA Main Window.

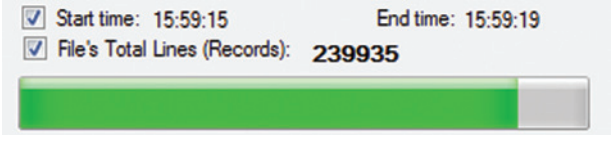

Figure 4. Analysis Process Progress Bar.

NsGTFA has the ability to read, analyse, and generate performance comparison for multiple trace files at the same time using tab pages as shown in Figure 6, and also graphs, as shown in Figures 7 and 8.

The application has many graph capabilities. You can draw packet delivery ratio, throughput, normalisation, overheads, end-to-end delay, and total dropped packet graphics. Also, the user can save performance data sets in a comma separated value (CSV) file format if they wish to plot them in any other different tools such as Gnu plot, Excel, etc.

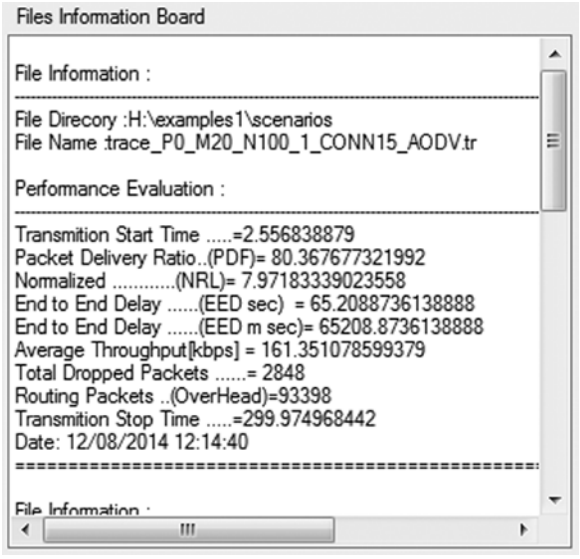

Figure 5. Results Report Figure.

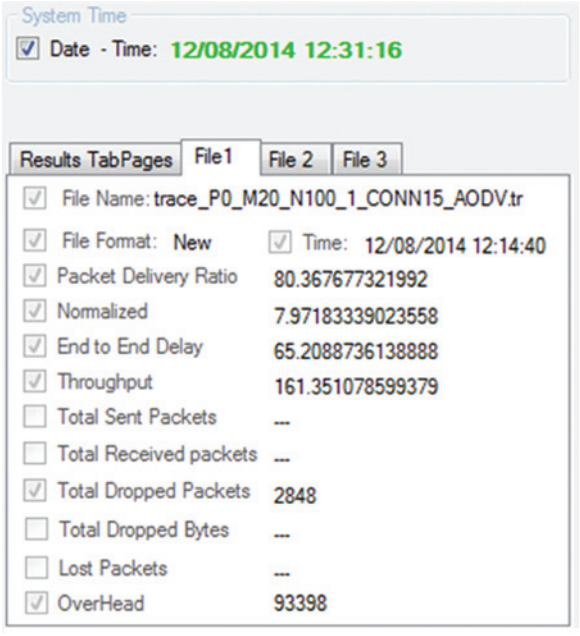

Figure 6. Results Report Tab Pages.

8 Appearance and Charts

To satisfy the user requirements, NsGTFA has many features through a friendly GUI and the ability to produce different types of charts (points, lines, and bars) in 2D and 3D formats in different colours and appearance, as shown in Figures 7–9.

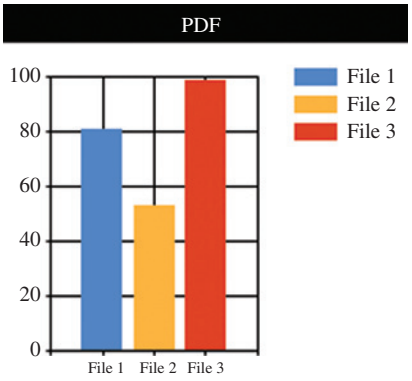

Figure 7. PDF Chart in 2D Bar Format.

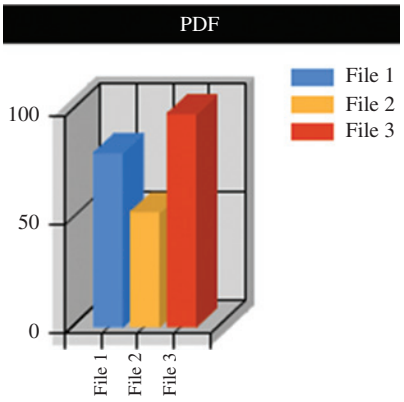

Figure 8. PDF Chart in 3D Bar Format.

9 NsGTFA Performance

In this section, we present some results regarding the processing time of NsGTFA compared with other tools when it comes to reading and analysing new trace files of various sizes. NGTFA is reliable and fast to read, and analyses a 60,107.254 byte file (~239,935 lines) in 51 s and a 1.2 GB file (4,948,948 lines) in 4.1 min, as shown in Table 1. It is obvious, however, that the increase of processing time is near linear to the trace file sizes, as one would expect.

10 Conclusion and Future Work

In this article, we presented a tool (NsGFTA) for analysing the output of Ns2 simulations of networks. Trace files of any size and of any of the standard formats are processed in a few seconds to produce standard performance metrics. The user can easily view and save these performance outputs as plots in 2D or 3D with user-selected colour coding.

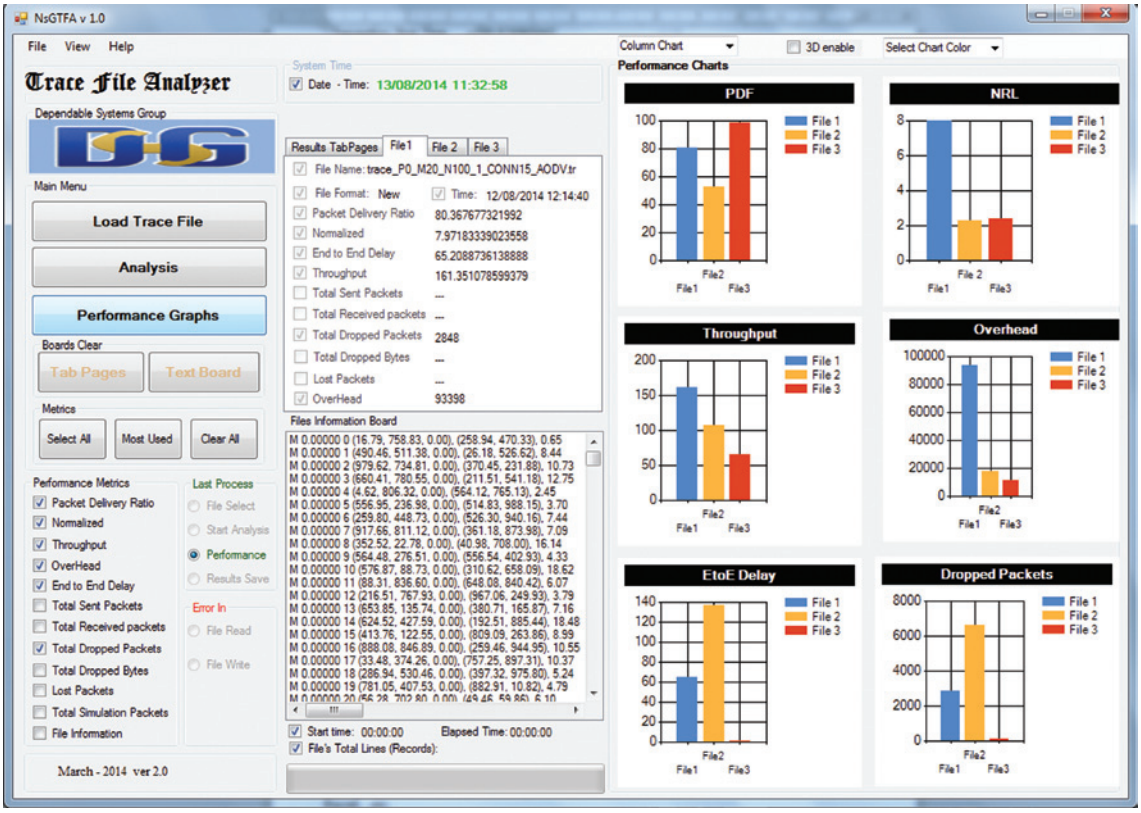

Figure 9. NsGTFA Appearance and Charts.

Table 1. Processing Time Results.

Trace File Size	Tool	Execution Time
2 MB	Jtarana	10 min
20 MB		100 min
60 MB		300 min
1.2 GB		6144 min
2 MB	TRAFIL	2.5 s
20 MB		25.36 s
60 MB		66.36 s
1.2 GB		18.4 min
2 MB	NsGTFA	1.7 s
20 MB		17 s
60 MB		51 s
1.2 GB		4.1 min

NsGFTA means that users of Ns2 do not need to produce their own trace analysers. Used in teaching and research, initial results can be produced without any need to understand the trace file format or develop scripts to process these files. Metrics such as packet delivery fraction, throughput, end-to-end delay, dropped packets, etc., are all immediately available.

Given the success of NsGFTA, we intend to upgrade it to calculate further metrics, such as transmission capacity and power consumption. We also intend to convert it to Java to produce a version that can be run under different operating systems.

The NsGTFA source code is available online at http://www.macs.hw.ac.uk/~isi3/.

Bibliography

[1] C. Bouras, S. Charalambides, M. Drakoulelis, G. Kioumourtzis and K. Stamos, A tool for automating network simulation and processing tracing data files, *Simul. Model. Pract. Theory* **30** (2013), 90–110.

[2] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, Varadhan K., Y. X. Y. Xu and H. Y. H. Yu, Advances in network simulation, *Computer (Long Beach Calif.)* **33** (2000), 59–67.

[3] V. L. Chee and W. C. Yau, Security analysis of TORA routing protocol, in: *Computational Science and Its Applications – ICCSA 2007, Pt. 2, Proceedings*, vol. 4706, pp. 975–986, 2007.

[4] C. Cicconetti, E. Mingozzi and G. Stea, An integrated framework for enabling effective data collection and statistical analysis with NS-2, in: *Proceeding from 2006 Work. NS-2 IP Network Simulator – WNS2 '06*, p. 11, 2006.

[5] K. Fall and K. Varadhan, The ns Manual (formerly ns notes and documentation), *VINT Proj.* (2011), 434.

[6] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 2nd ed., p. 527, Springer, New York, 2012.

[7] A. Kaponias, A. Politis and C. Hilas, *Simulation and Evaluation of MANET Routing Protocols for Educational Purposes*, in: Proceedings of the 2nd Pan-Hellenic Conference on Electronics and Telecommunications, Thessaloniki, pp. 2–5, 2012. Available: http://www.pacet.gr/program.htm.

[8] NS2 Analyser: Jtrana, [Online] Available: http://ns2analyser.blogspot.co.uk/2011/03/jtrana.html, Accessed 6 May, 2014.

[9] OTcl – MIT Object Tcl, [Online] Available: http://www.isi.edu/nsnam/otcl/README.html, Accessed 6 May, 2014.

[10] V. Park and S. Corson, Temporally-ordered routing algorithm (TORA) version 1, *IETF MANET Work. Gr.*, pp. 1–24, 2001.

[11] F. Rocha, *NS2 Visual Trace Analyzer*, *Manual*, p. 17, 2010.

[12] A. U. Salleh, Z. Ishak, N. M. Din and M. Z. Jamaludin, Trace Analyzer for NS-2, in: *2006 4th Student Conf. Res. Dev.*, 2006.

[13] S. Taruna and G. N. Purohit, Scenario based performance analysis of AODV and DSDV in mobile ad hoc network, *Advances in Networks and Communications, First International Conference on Computer Science and Information Technology*, 2011, [Online] Available: http://download.springer.com/static/pdf/982/chp%253A10.1007%252F978-3-642-17878-8_2.pdf?auth66=1399550937_f2886aebb968ba9ad3d23b20601ab1a4&ext=.pdf, Accessed 6 May, 2014.

[14] VINT Project, [Online] Available: http://www.isi.edu/nsnam/vint/index.html, Accessed 6 May, 2014.