OPNET Modeler and Ns-2: Comparing the Accuracy Of Network Simulators for Packet-Level Analysis using a Network Testbed

Gilberto Flores Lucio, Marcos Paredes-Farrera, Emmanuel Jammeh, Martin Fleury, Martin J. Reed
Electronic Systems Engineering Department
University of Essex
Colchester, Essex C04 3SQ
United Kingdom
gflore@essex.ac.uk, http://privatewww.essex.ac.uk/~gflore/

Abstract: - This paper presents a comparative study of two well-known network simulators: OPNET Modeler and Ns-2. Other studies in this area have generally been confined to just one simulator. The motivation of this paper is to provide a guide to researchers undertaking packet-level network simulations. The simulator outputs were compared to the output from a live network testbed. The experimental comparison consisted of deploying both CBR data traffic, and an FTP session, both on the network testbed and the simulators. CBR data traffic was used due to its simplicity of modeling. A custom analysis tool was employed to examine the behavior of the network in different background traffic scenarios. The same scenarios were then recreated in the simulators in order to ascertain their realism. The results show the necessity of fine-tuning the parameters within a simulator so that it closely tracks the behavior of a real network.

Keywords: network simulator, OPNET Modeler, Ns-2, simulation methodology

1 Introduction

Network simulators have grown in maturity since they first appeared as performance, management and prediction tools. Simulators are normally used as network management tools, for which packet level analysis is not commonly employed. However, more studies are needed to establish guidelines for researchers so that they may select and customise a simulator to suite fine-grained packet level analysis [1][2]. Reference [3] reports `the breach of credibility' that studies based on simulation tools need to tackle; one motivation behind this paper is to address this need. The ease and facility that simulators provide in evaluating "radical" changes to a network environment cannot be discarded. There are a considerable number of simulations tools in the market. The main characteristics that divide them are: accuracy, speed, ease of use, and monetary expense.

This paper concentrates on the accuracy of the simulation in comparison to a real network for packet level analysis. Two of the currently popular network simulators that can perform this type of analysis are Modeler from OPNET[1] [4] and Ns-2 from the Virtual Internetwork Testbed project VINT [5]. These are selected because of their popularity within academia, commercial and industrial communities [6].
An ideal way to test simulator accuracy is to measure real network traffic and compare it with the simulators results. Therefore, this paper presents real and simulated results (from the two simulator tools), where CBR (Constant Bit Rate) data traffic, and an FTP (File Transfer Protocol) session are generated in order to test the network performance with different types of traffic. CBR traffic was selected due to the simplicity of its nature and FTP due to its popular use and dynamic behavior. CBR in particular is a useful analysis as it provides "ground truth" *i.e.* there are few external factors that influence the model. Whereas FTP was chosen at this stage as any more complex protocol (*e.g.* HTTP) would have too many degrees of freedom. The analysis is made at a packet-by-packet level concentrating on the bandwidth utilisation analysis.

The initial tests used the default configuration profiles provided with the simulators, however, several manual modifications were made at different stages of the experiments to investigate possible improvements. An in-house analysis tool called `tcpflw` was used to examine the behavior of the real network. `tcpflw` has successfully been used to analyze the performance of a video streaming session [7]. The results reported here were directly compared with the results from the simulators.

This paper does not aim to define which network simulator is best as there are too many different parameter variations and different possible network scenarios to adequately determine this in a single paper. Instead, this paper demonstrates how the suitability of simulators can be validated for the particular case of packet level forwarding in an IP

[1] OPNET Modeler was provided under the OPNET University Programs

transport network supporting both real-time CBR and non-real time data transport. It is hoped that it will act as a guideline to other researchers of the validation steps required when choosing and optimising a network simulator for a different scenario.

The rest of the paper is organized as follows: Section 2 presents the network simulators, the experimental testbed and experiments used for the tests. Section 3 presents the comparative results obtained. Section 4 gives a constructive discussion of the difficulties encountered with each simulator. Finally, in Section 5, conclusions are drawn, and further work is proposed.

2 Overview of the experimental setup

Network simulators can be divided into several types (by protocol, technology or processing method), but the most general categorization is their method of simulation. There are two typical methods of simulation: discrete event or analytical simulation [8]. The former produce predictions in the network at a low level (packet-by-packet), which makes them accurate but slow to generate results. The latter, use mathematical models to produce their results at a much faster speed, but may sacrifice accuracy. The usual approach is to combine both methodologies with the aim to provide a reasonable performance in terms of speed but maintaining accuracy in the critical areas. The two simulators used in these experiments are hybrid simulators.

2.1 The simulation tools

This Section gives a brief description of the capabilities of each simulator.

OPNET Modeler The Modeler is just one of the many tools from the OPNET Technologies suite. The version used for these experiments was 9.0.A. The engine of the OPNET Modeler is a finite state machine model in combination with an analytical model. Modeler can model protocols, devices and behaviors with about 400 special-purpose modelling functions [9]. The GUI (Graphical User Interface) along with the considerable amount of documentation and study cases that come along with the license are an attractive feature of the Modeler. A number of editors are provided to simplify the different levels of modelling that the network operator requires. Modeler is not open source software though model parameters can be altered, and this can have a significant effect on the simulation accuracy.

Ns-2 is the second version of a network simulator tool developed by the Virtual InterNetwork Testbed (VINT) project [10]. It is an event-driven network simulator, which is popular with the networking research community. It includes numerous models of common Internet protocols including several newer protocols, such as reliable multicast and TCP selective acknowledgement [11]. A network animator, `Nam` [12], provides packet-level animation and protocol specific graph for design and debugging of network protocols. Additionally, different levels of configuration are present in Ns-2 due to its open source nature, including the capability of creating custom applications and protocols as well as modifying several parameters at different layers.

2.2 The network testbed

Fig. 1 shows the network testbed used for these experiments. It is formed by five PC's, each with processor rating of 1.7 GHz, running the Linux operating system with IntelPro 400 network cards. Two of them operate as a client-server pair, and the other two as a traffic-generator/traffic-sink pair (Traffic G. and Traffic S. respectively), and the fifth acts as a router. In addition, a 10 Mbit hub is used to connect two PC's to the Router. Links of 10 and 100 Mbit/s were employed. The Linux systems had a kernel patch applied (Kansas University Real-Time Linux, KURT [13]) to reduce the scheduling granularity. This was required so that the scheduling of high-rate CBR traffic (of the order 5 Mbit/s used in the experiments) could be maintained with reasonably low delay variation (jitter). Without this patch the performance was severely degraded due to scheduler problems.

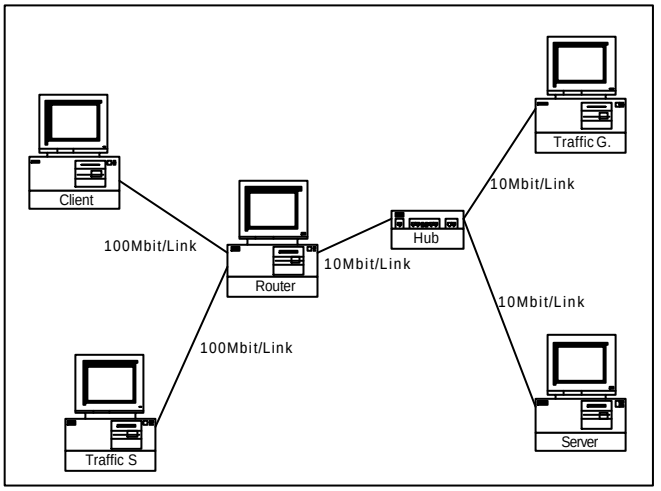

Fig. 1 Network testbed.

2.3 Network representation in the simulators

OPNET Modeler provides different levels of modeling depending on the necessities and requirements of the simulation. The GUI establishes an overall environment called a *Project* [4]. From that Project, the operator can develop several network scenarios in order to evaluate and analyze the

performance of that network in different "what-if" circumstances. Fig. 2 shows the network representation used in OPNET for these experiments. Two separate scenarios were used. The first is used for the CBR experiment, and the second for the FTP session. Both representations contained the same elements; with the difference that the former was created from custom process levels to simulate more accurately the real characteristics of the network testbed and the latter uses the standard and a custom tuned version of FTP.

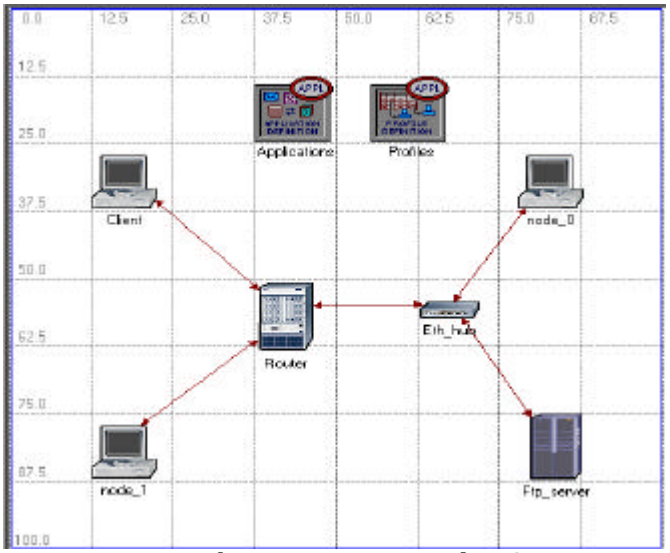

Fig. 2 Network representation used in OPNET Modeler.

The Ns-2 architecture takes an object-oriented approach using `C++` and `otcl` [14]. `Otcl` is an object-oriented variant of the well-known scripting language tcl. `C++` is utilized for per-packet processing and forms the core of the simulator. `Otcl` is used for simulation scenario generation, periodic or triggered action generation and for manipulating existing `C++` objects. Fig. 3 shows the network representation for Ns-2 for these experiments.

2.4 The experiments

Two sets of experiments were conducted. The first one was to test the simulators accuracy with raw CBR data traffic and the second with an FTP session to simulate best-effort data traffic:

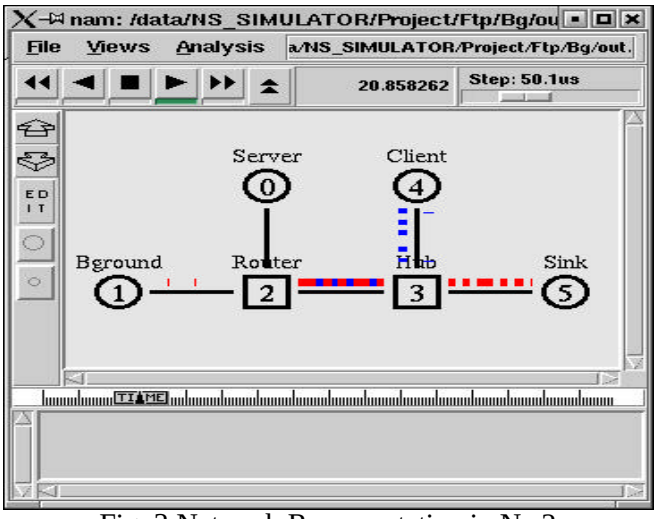

Fig. 3 Network Representation in Ns-2.

CBR traffic is characterized by a fixed bandwidth across the network and is typically used by applications such as video and audio. Furthermore, these types of applications usually require strict delay and jitter bounds on the CBR packets used for transport. A CBR traffic stream can be generated by fixing the packet size and using the same inter-arrival time between these packets.

This type of traffic was produced by introducing trace files into the simulators tools. The traffic generators, *stg* and *rtg*, which come with the *NCTUns* [15] simulator tool were used in the network testbed. The *stg* program can read from a trace file and replicate the UDP (User Datagram Protocol) packets.

FTP session. FTP is intended to share, transfer and transmit information between two computers. Bhushan [16], in the MIT Project MAC, proposed the first file transfer mechanism in 1971. Nearly all of the FTP servers and clients follow the more recent RFC no. 959 [17], in which is explained: the usage model for FTP; the set of commands employed; and how the protocol handles the control commands and the data connection.

For the FTP experiments, two types of tests were performed:

1. -FTP sessions using the default settings of the simulators.
2. -Finely tuned FTP parameters at different levels for each simulator capability. The actual parameters changed are discussed next.

Several tests were performed to evaluate the tuned parameters and the values that better "mimic" the characteristics of the networks testbed for each simulator. The most significant parameters tuned in each simulator with the default and "tuned" values are given in Table 1. For OPNET Modeler, three parameters were changed, first the version of TCP was

changed from *Reno* to *New Reno*; this improves the fast recovery capability of the protocol. When multiple packets are lost, New Reno can recover without a retransmission timeout [18]. *Window Scaling* was activated, this allows the advertisement of window sizes bigger than 65 kB [19], and, the *Time-Stamp* [19] option was activated to imitate the echoing capability of the testbed in both directions. For Ns-2 [11], the *link latency* was activated and this affects the Round Trip Time (RTT) of the packets. *Max segment size* was increase for bigger Ethernet packets, and *Window size* to have more range for sliding the window value. With Ns-2 there was no discernable change between *Reno* and *New Reno*.

Simulator	Parameters	Default/tuned value
OPNET Modeler	New Reno	Disabled/Enabled
	Window Scaling	Disabled/Enabled
	Timestamp	Disabled/Enabled
Ns-2	Link Latency	Disabled/Enabled
	Max segment size	1000/1500
	Window size	20/100

Table 1. Parameters tuned for both simulators.

2.5 The methodology

To evaluate the accuracy of the simulators, actual network data was measured. Two sets of experiments were conducted with different scenarios. First, CBR data traffic was introduced to the network testbed and second, a FTP session was performed from a client to a server to transfer a 10 MB file. Under all the scenarios, the measurement technique consisted in monitoring the IP traffic using `tcpdump` [20], which is a tool widely used for packet capture in the research community [21]. The information provided for every packet was: measurement timestamp, source/destination IP address and port source, Ethernet packet size, and IP packet size.

Afterwards, the data were analyzed with the in-house program `tcpflw`, obtaining the Ethernet throughput in every node in the testbed. `tcpflw` is software developed in the University of Essex by Marcos Paredes-Farrera that groups and analyzes IP data obtained from `tcpdump`. The data consists of packets that are grouped in different categories called flows. Every flow can be characterized by its IP source, IP destination, port source/port destination, and protocol. The program can provide graphs and trace files of every flow by performing additional post-processing of the raw data. The granularity can be changed to analyze the traffic from an application, service, and computer, at subnet and network level. The program can analyze packet size, bandwidth, and inter-arrival time versus time. For this study, bandwidth analysis was used.

In total, 24 experiments were conducted with 8 different scenarios. 12 experiments were performed for CBR traffic and 12 for the FTP session. Table 2 shows the different scenarios, some with cross-traffic and some without. Each scenario was repeated in the two simulators and in the network testbed.

Scenario #	Client-Server Load	Traffic G. - Traffic S. load
CBR1	2Mb/s	0 Mb/s
CBR2	2Mb/s	2 Mb/s
CBR3	5Mb/s	0 Mb/s
CBR4	5Mb/s	6Mb/s
FTP1	10MB file	0Mb/s (CBR)
FTP2	10MB file	6Mb/s (CBR)
FTP3	10MB file	0Mb/s (CBR)
FTP4	10MB file	6Mb/s (CBR)

Table 2. The CBR and FTP traffic scenarios.

3 Results

Figs.4–8 compare the most relevant results obtained by the two simulators and the network testbed for the first four CBR scenarios. Fig. 4 shows the results obtained in scenario CBR1, from the router's perspective. As can be seen, Ns-2 displays a more realistic CBR behavior than Modeler. We surmise that the reason for this could the method used to import CBR traffic pattern into Modeler. There are three techniques to import this traffic into Modeler:

1. - Import traffic measurements as trace files using the OPNET ACE flow analysis tool.
2. - Create a custom process model that imports trace files.
3. - Import trace files as a background traffic source within OPNET.

We have used the latter as OPNET ACE flow analysis was not available to us and generating a custom process model was not required for Ns-2. Modeler used the trace file as a source to generate background traffic at the given rate during this part of the experiment, which no doubt explains the “bursty behavior” [22] displayed.

Figs. 5 and 6 (scenarios CBR2 and CBR3) show very similar simulator behavior compared to the CBR1 behavior.

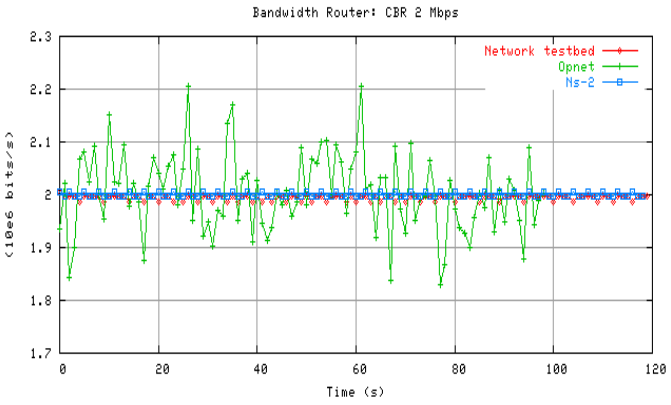

Fig. 4 CBR1 Router perspective.

Figs. 5 and 6 show lower throughput for the network testbed. This is in fact an artifact of the scheduling problems mentioned in Section 2.2; even with the applied KURT kernel patch a target CBR rate of 5 Mbit/s was not quite achieved, and without the patch this rate was reduced much further.

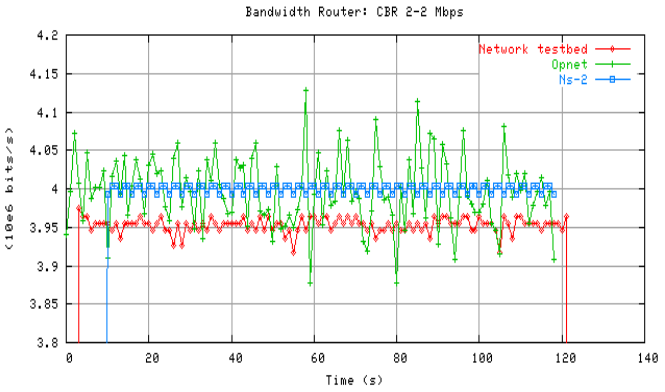

Fig. 5 CBR2 Router perspective.

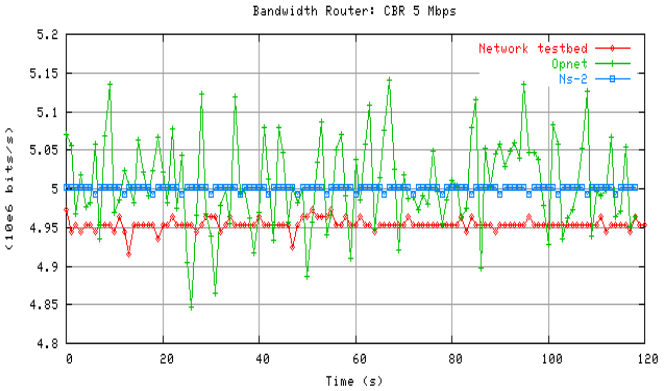

Fig. 6 CBR3 Router perspective

Figs.7–9 show results from scenario CBR4. Again, the testbed traffic discrepancy when generating CBR traffic appears in Figs. 7 and 9. Turning to the simulators, Fig. 7 shows a sudden drop in bandwidth in Ns-2 from the client perspective. The sudden drop in bandwidth came as a surprise, so this result was repeated a number of times to confirm the anomalous problem. In scenario CBR4, the Modeler from OPNET provides a more accurate behavior in comparison to the network testbed than the other simulator.

It was found that in the real network testbed the traffic from different flows was apportioned unequally by the router packet scheduler, whereas both simulators split the flows equally in the router. Clearly, the simulators have made an assumption about the router scheduling, whereas, in practice, a number of different algorithms may be used in the router kernel. This shows one weakness in packet level simulation, in that, actual systems have a great number of variables that control the forwarding of packets and not all of these processes can be accurately replicated.

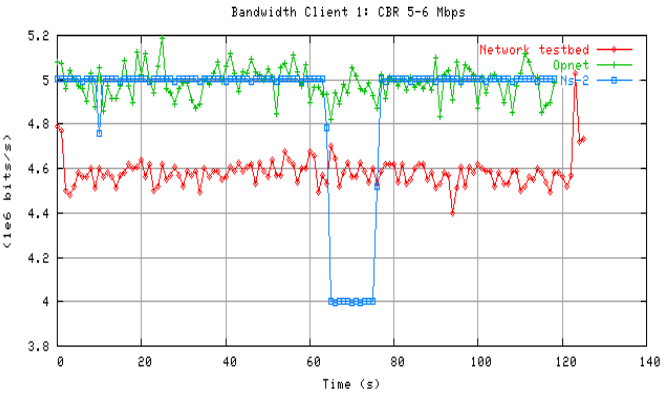

Fig. 7 CBR4: Client perspective.

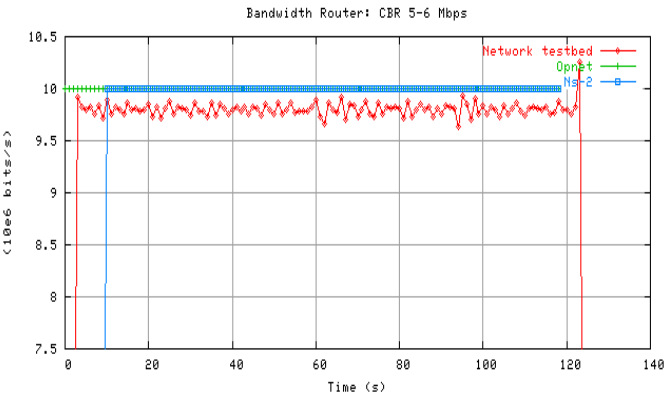

Fig. 8 CBR4: Router perspective.

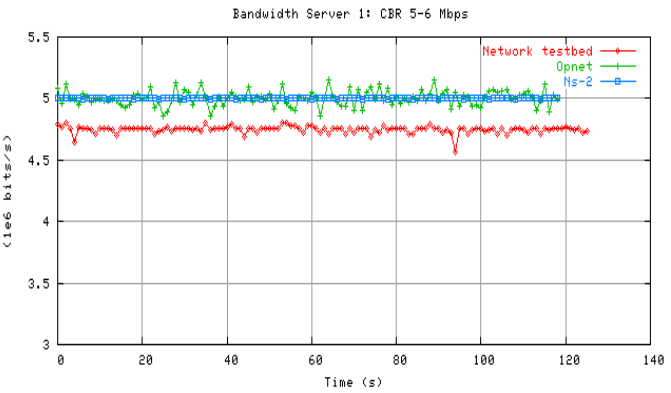

Fig. 9 CBR4: Server perspective.

Figs.10–13 compare the most relevant results obtained by the two simulators and the network testbed for the two FTP scenarios with and without the fine-tuning. In both scenarios, a 10 MB file was

transferred from the Server to the Client. Traffic going from client to server consists of session level packets and TCP data transfer acknowledgements. Figs. 10 and 11 show the results for scenarios FTP1 and FTP3 from the router perspective with no CBR traffic the former is the "standard" version of FTP and the later with the tuning parameters active. For scenarios FTP2 and FTP4, CBR traffic of 6 Mbit/s was also sent from G to S (see Fig. 1 for node definition). The results displayed only show the file transfer traffic; they do not consider the preliminary traffic to establish the FTP session and the subsequent traffic to tear the connection down. The FTP used in the network testbed was vsFTPd version 1.1.0. [2]

Though vsFTPd has more security features than some other versions of FTP, its traffic behavior appears to be typical. From the results, we can conclude that to model a FTP server is not an easy task, an observation echoed by others [23]. While Ns-2 took more time to transfer the file due to its limited and almost constant bandwidth (as shown in Figs. 10 and 11), Modeler had a tendency to more closely model the network testbed over time. A simple FTP session, due to its dynamic behavior and number of interrelated factors, is a difficult application to simulate. Nevertheless, after "fine-tuning" simulator parameters, the results from the Modeler were quite accurate in comparison to those from Ns-2. In particular, note the behavior of the FTP3 session of Modeler shown in Figs. 10 and 11 matches the actual network testbed quite closely.

Figs. 12 and 13 show the results for scenario FTP2 and FTP4 from the router perspective. Again, the former represents the standard FTP version of the simulators and the second the "fine-tuned" one. The results show significantly different tendencies between the Modeler and Ns-2. The OPNET Modeler is a more accurate replication of the testbed results.

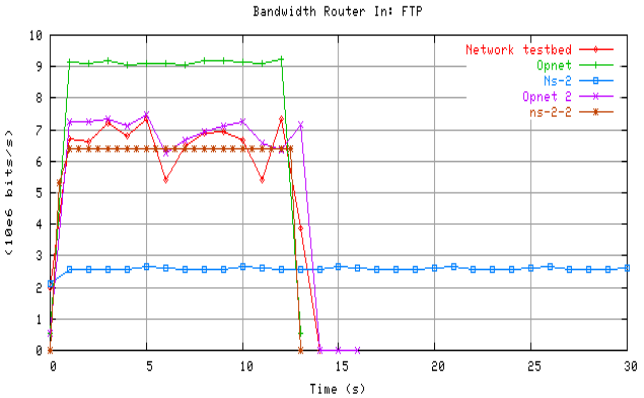

Fig. 10 FTP1 and FTP3: Router perspective (from Server to Client).

[2] vsFTPd is available at http://vsftpd.beasts.org/ (current v 3 03).

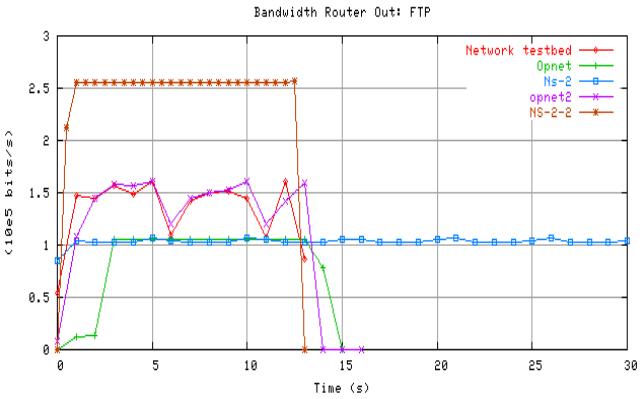

Fig. 11 FTP1 and FTP3: Router perspective (from Client to Server).

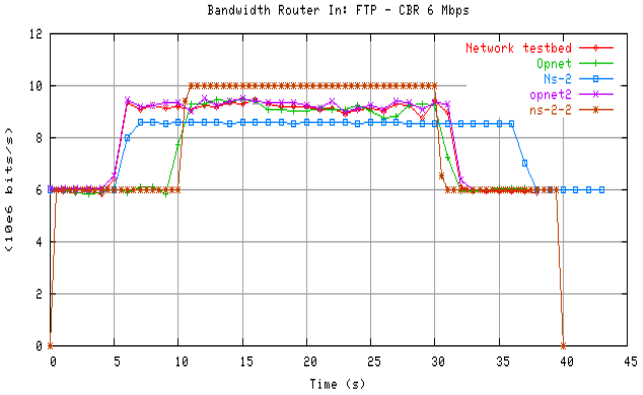

Fig. 12 FTP2 and FTP4: Router perspective (from Server to Client).

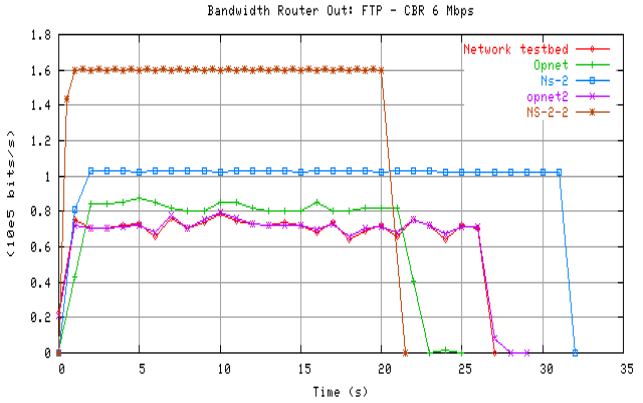

Fig. 13 FTP2 and FTP4: Router perspective (from Client to Server).

4 Discussion

In terms of accuracy of bandwidth estimation for the pure CBR-type traffic, Ns-2 performed better than OPNET Modeler using the default Modeler package. This may be improved using additional OPNET packages; the authors cannot verify this. Nevertheless, in scenario CBR4, Ns-2 behaved differently to the testbed, and instead Modeler gave more accurate results.

Turning to the FTP experiments. The Ns-2 FTP simulation model only indicated a general transfer rate rather than replicating the actual network flow.

Modeler performed closely to the testbed results, and in the case of the FTP2 and FTP4 scenarios the results were remarkably similar.

In terms of simulation speed, both Ns-2 and the Modeler proved to be fast, requiring less than a minute to obtain the results.

Ns-2 has a "small suite", but for large-scale networks several modifications and extra care has to be taken to manage memory allocation and CPU time (abstraction techniques)[24]. OPNET Modeler has a "heavy suite" (large software overhead) but provides diverse statistics modules at different levels [25]. The freeware nature of Ns-2 is attractive to researchers compared to the need to enter into an OPNET Modeler license agreement and associated direct costs. Additionally, the constraint of not having access to other modules outside the standard version of OPNET, such as the Application Characterization Environment (ACE) and the SPGuru package, limits the possibility of evaluating other performance measures such as flow-analysis.

At first glance, the evaluation of the different network simulators by comparison with the testbed appears to be a straightforward procedure. However, this was not the case. The learning curve for each of the simulators was different, and sometimes steeper than expected. To create an Ns-2 model involves writing a script in an extension of `tcl`, which will be unfamiliar to most of those using the simulator for the same time. On the other hand, as benefits a commercial product, OPNET has a well-engineered user-interface using mainstream software and operating system.

5 Conclusions and further work

In this paper, two network simulators were compared with a network testbed. The accuracy of Ns-2 and Modeler from OPNET was compared using CBR data traffic and an FTP session. Several scenarios were evaluated and regenerated in the simulation tools and the network testbed. The results provide interesting guidelines to network researchers in the selection of network simulation tools.

From the researchers point of view, Ns-2 provides very similar results compared to OPNET Modeler, but the "freeware" version of Ns-2 makes it more attractive to a researcher. However, the complete set of OPNET Modeler modules provides more features than Ns-2, and it therefore will be more attractive to network operators.

One specific observation can be made about these network simulators as a result of these experiments. For a simple CBR data traffic, it appears that the simulators had no significant problem in terms of accurately modeling the testbed behavior. However, in the case of an FTP session, the simulators using default simulation settings did not adequately model the dynamic behavior of FTP when is used in its basic standard form. FTP (through TCP flow control) adapts its output to prevailing network conditions, whereas the response of Ns-2 and OPNET Modeler did not always mimic this performance. However, when "fine-tuning" of parameters was performed, it was found that Modeler was a more accurate simulator for this particular case.

We did not find creating these types of comparisons between different network simulation tools to be an easy task. Further work to be performed includes establishing a scheme to model HTTP and other more complex protocols in the simulators and the testbed.

References:

[1] R. Currier. Test-drive your network designs. *Network World*, May 1999.

[2] J. Heidemann and K. Mills. Expanding confidence in network simulations. *IEEE Network Magazin*e, 15(5): 58-63, 2001.

[3] K. Pawlikowski, H-D. J. Jeong, and J-S. R. Lee. On the credibility of simulation studies of telecommunication networks. *IEEE Communications Magazin*e, 40(1): 132-139, 2002.

[4] C. Zhu, O. W. W. Yang, J. Aweya, M. Oullette, and D. Y. Montuno. A comparison of active queue management algorithms using the OPNET Modeler. *IEEE Communications Magazin*e, 40(6): 158-167, 2002.

[5] V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *Winter Simulation Conferenc*e, pages 1037-1044, 1997.

[6] L. Breaslu, K. Estrin, D. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, X. Ya, and H. Yu. Advances in network simulation. *IEEE Compute*r, 33(5): 59-67, 2000.

[7] E. Jammeh, M. Paredes, and M. Ghanbari. Transporting real time transcoded video over the Internet using end-to-end control. In *International Packet Video Worksho*p, 2002.

[8] R. Currier. Test-drive your network designs. *Network World*, May 1999.

[9] *OPNET Users' Manual*, *OPNET Architecture*, *OV*.415.http://forums.opnet.com.

[10] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, M. Haldar, P. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejajie, S. Punnet, K. Varadhan, X. Ya, H. Yu, and D. Zappala. Improving simulation for network research. Technical Report 99-702b, USC Computer Science Department, 1999.

[11] K. Fall. Network emulation in the VINT/ns simulator. In *Fourth IEEE Symposium on Computers and Communications (ISCC'99*), pages 59-67,1999.

[12] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, and H. Yu. Network visualization with the VINT Network Animator Nam. Technical

Report 99-703b, USC Computer Science Department, 1999.
[13] W. Dinkel, D. Niehaus, M. Frisbie, and J. Woltersdorf. *KURT Linux Users Manual*, 2002. http://www.ittc.ku.edu/kurt/papers/user-manual-DRAFT.pdf.
[14] *The Otcl Tutorial*, 1995. http://bmrc.berkeley.edu/research/cmt/cmtdoc/otcl/tutorial.html (Current v 3 03).
[15] S.Y. Wang and H. T. Kung. A new methodology for easily constructing extensible and high fidelity TCP/IP network simulators. *Computer Networks*, 40(2): 257-278, 2002.
[16] A. Bhushan. A file transfer protocol, Internet Engineering Task Force RFC 114. Technical report, 1971.
[17] J. Postel. File transfer protocol (ftp), Internet Engineering Task Force RFC 959. Technical report, 1985.
[18] The New Reno Modification to TCP's Fast Recovery Algorithm, Internet Engineering Task Force RFC 2582.
[19] TCP Extensions for High Performance, Internet Engineering Task Force RFC 1323. pages 9-11.
[20] `tcpdump` manual pages. http://www.tcpdump.org/tcpdump man.html.
[21] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. dissertation, U.C. Berkeley, 1997
[22] Organizing server assets: Methodology for organizing server assets, 2003. From the OPNET Methodology and Case Studies Website, part of the Technical Resources papers.
[23] M. Masuda and J. Ichimura. Evaluation of simulation accuracy and network planning case study. OPNET website white paper.
[24] The Network Simulator ns-2: Tips and Statistical Data for Running Large Simulations in NS; http://www.isi.edu/nsnam/ns/ns-largesim.html.
[25] W. G. Bragg. Which network design tool is right for you? *IEEE IT Pro Magazin*e, 2(5): 23-31,2000.