Analysis and experimentation of Internet Traffic Generator

Stefano Avallone, Antonio Pescapè, Giorgio Ventre

Abstract— **Network researchers have dedicated a notable part of their efforts to the area of modeling traffic and in particular to the definition of rigorous statistical models. We feel that there is a strong demand for traffic generators capable to reproduce realistic application traffic according to theoretical models. Indeed, nowadays the trend is changing and network researchers are melting together networking and stochastic knowledge in order to fill in the actual gap. This work presents an integrated set of tools we called Internet Traffic Generator (ITG), which allows to reproduce TCP and UDP traffic and to accurately replicaie appropriate stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random processes. We believe that ITG shows interesting properties when compared to other traffic generators.**

Keywords— **Network Traffic Generator, Network Traffic Models, Network Performance Evaluation**

I. Introduction

As computer networking has become more ubiquitous, researchers are increasingly focused on optimizing computer networks performance and improving network utilization in terms of throughput and offered delay, jitter and packet loss. This process cannot leave the study of traffic patterns and properties out of consideration. In the last twenty years researchers have been looking for the definition of stochastic processes that could be used as accurate and simple models for traffic generation in packet switched networks and in particular in IP networks. In order to be as realistic as possible, traffic models should accurately represent relevant statistical properties of the original traffic [1]. Modeling the Internet traffic is an important and essential task and we think that traffic theory should be increasingly used to guide the design of the future multi-service and integrated Internet. It is unlikely that we will be able to understand the traffic characteristics, predict network performance (Quality of Service (QoS), Service Level Agreement (SLA) definition,...), or design dimensioning tools without analytical and rigorous models. The successful evolution of the Internet is tightly coupled to the ability of designing simple and accurate models with the property of reproducibility. Traffic theory suggests us the application of mathematical modeling to explain the relationship between traffic performance and network capacity, traffic demand and experimented performance.

Network management has so far been dominated by passive monitoring. Emerging networking technologies however force the development of active testing and performance analysis tools. In the case of studies related to the Internet, the experiments should not only reflect the wide scale of real scenarios, but also the rich variety of traffic sources, in terms of both protocol typologies and data generation patterns. As a consequence, traffic models can be applied to the generation of synthetic, yet realistic traffic to be injected into a network. For this purpose, we developed a tool, named ITG (Internet Traffic Generator) that generates network traffic according to the models proposed for different protocols. We implemented several protocols belonging to layers from 4 to 7 of the ISO stack. The user can simply choose a protocol and is not requested to know its model. In addition, the user can generate a specific traffic pattern by using several random distributions to model the IDT (Inter Departure Time) and PS (Packet Size) processes.

The rest of the paper is organized as follows. Section II presents the motivation of our work and a survey on related work. Section III provides general details and software architecture aspects of our Internet Traffic Generator. Section IV shows some usage examples of our generator and the results achieved. Section V discusses the open issues and the perspective for future work.

II. Motivation and Related Work

The purpose of our Internet Traffic Generator is to build up a suite that can be easily used to generate repeatable sets of experiments by using a reliable and realistic mixture of traffic typologies. ITG enables to generate many traffic scenarios that could be originated by a typical network test-case made of a large number of users and network devices, as well as by different network topologies. Our generator can simulate (and not emulate) traffic. For traffic simulation we mean the reproduction of a "traffic profile" according to theoretical stochastic models. Instead, for traffic emulation we mean the reproduction of a specific protocol (i.e. reproduction of http messages without using a browser). The generation of realistic traffic patterns can help in understanding the protocols and applications of interest in today's Internet. ITG can generate UDP and TCP traffic and is designed for the generation of "layer 7" traffic (application layer traffic). ITG primary design goals are:

- reproducibility of network experiments: exactly the same experiment can be repeated several times by choosing the same seed value for the packet interdeparture and packet size random processes
- investigation of scaling effects: scalability problems can be investigated by using different network loads or different network configurations

S. Avallone, A. Pescapè and G. Ventre are with the Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Italy. E-mail: {*stavallo, pescape, giorgio*}@*unina.it*.

• improvement of generation performance with respect to other traffic generators
• comparison of practical, analytical and theoretical evaluations based on mathematical and statistical methods

Through the use of our tool, a network administrator can evaluate the performance of a network, locate possible problems and trace guidelines for network planning and real implementation. The outcome of our work was a software tool available to network researchers and designers who need a scientific way to prototype new applications and protocols in a real testbed with realistic traffic.

We now provide an overview of some of the most widely used traffic generators [2].

TG Traffic Generator [3] runs on Linux, FreeBSD and Solaris SunOS. TG is capable to generate constant, uniform, exponential on/off UDP or TCP traffic. TG does not offer a rich variety of traffic sources.

NetSpec [4] is a traffic generator/emulator that allows the user to define multiple traffic flows from/to multiple computers. It is capable to emulate TCP, UDP, WWW, FTP, MPEG, VBR and CBR Traffic. Netspec runs on Linux, FreeBSD, Solaris and IRIX.

Netperf [5] provides tests for both unidirectional throughput and end-to-end latency. The environments currently measurable by netperf include TCP and UDP via BSD Sockets, DLPI, Unix Domain Sockets and Fore ATM API.

Packet Shell [6] is an interactive shell and active traffic generator. It represents the "Packet Shell", an extensible Tcl/Tk based software toolset for protocol development and testing. The Packet Shell creates Tcl commands that allow the users to create, modify, send, and receive packets on networks. The operations available for each protocol are supplied by a dynamic linked library called a protocol library. Packet shell runs on SunOS 5.X or later. The output is textual.

MGEN [7] is both a command line and GUI traffic generator. It runs on Linux, FreeBSD, NetBSD, Solaris, SGI and DEC. MGEN provides programs for sourcing/sinking real-time multicast/unicast UDP/IP traffic flows. The MGEN tools transmit and receive time-stamped, sequence numbered packets. The analyses of the log files can be performed to assess network or network component ability to support the given traffic load in terms of packet loss, delay, jitter, etc.

Rude/Crude [8] is a command line traffic generator and measurement tool for UDP. RUDE stands for Real-time UDP Data Emitter and CRUDE for Collector for RUDE. Currently these programs can generate and measure only UDP traffic. The operation and configuration might look similar to MGEN. RUDE/CRUDE tools were designed and coded because of the accuracy limitations of MGEN. Rude/crude runs on Linux, Solaris and FreeBSD.

ITG achieves performance comparable to that of RUDE/CRUDE but additionally it makes available a greater number of traffic source types.

UDPgen [9] is a command line UDP traffic generator integrated into the Linux kernel. It aims at maximizing the packet throughput especially for Gigabit Ethernet. To do this, the traffic generator runs completely in the Linux kernel. This allows sending at much higher rates than with an user space program. The toolset also includes a tool which counts UDP packets at the receiver and calculates the packet inter-arrival times.

Linux Traffic Generator [10] can generate multiple independent UDP flows with given traffic profiles (i.e. CBR or VBR), with millisecond resolution. LTG works on a common PC with Linux operating system. It is possible to evaluate a set of performance metrics related to throughput, loss and delay. The LTG tool derives from the TTCP tool [11]. LTG allows generating multiple independent flows of UDP traffic. LTG evaluates the average throughput and can log into a file the "instantaneous" throughput. This generator is not publicly available.

Traffic Generator (TG) [12] generates and receives one-way packet traffic streams transmitted from the UNIX user level process between traffic source and traffic sink nodes in a network. TG is controlled by a specification language that allows access to different operating modes, protocols, addressing functions, and experimentation with traffic parameters. The specification language allows traffic of several different packet lengths and inter-arrival time distributions to be generated. The current implementation supports TCP and UDP transport protocols, with unicast and multicast addressing (UDP only).

Traffic [15] generates high volumes of traffic on a network and does not measure throughput or response times. It has a friendly GUI and it runs on Microsoft Win32, FreeBSD and Linux. A limited set of traffic random variables is available.

PacGen [16] is an Ethernet IP TCP/UDP packet generating tool for Linux. It generates experimental ARP packets too. This tool enables custom packets with configurable Ethernet, IP, TCP, and UDP layers as well as custom payloads.

NTGen [17] (Network Traffic Generator) is a Linux kernel module (supports Linux kernel version 2.4.* and later) that generates network packets. The module sends these packets on the network interface. It supports common network protocol packets (ethernet, IP, TCP, UDP, ARP ...) and it uses a well-defined meta language (Bison & Lex) for configuring packet generation streams. An user-space application allows the user to configure plans (e.g. streams) of packets generation for the kernel module. We are focusing our attention on this module in order to further improve the performance of ITG.

After using some of the presented traffic generators in our network testing and network measurement operations, we experimented the lack of the necessary characteristics in a single traffic generator. Therefore we decided to implement our ITG. The basic idea of creating a new traffic generator arose from the lacks

of existing ones with the possibility of simulating a rich variety of traffic sources, reproducing an experiment by repeating many times exactly the same traffic pattern (not only its mean values) and getting information not only about received packets but also about transmitted packets. ITG has been planned for the generation of network traffic (ICMP), transport layer traffic (TCP and UDP), several "layer 5-7" traffic (HTTP, FTP, TELNET, SMTP, DNS, VoIP, Video, NNTP, ...).

Finally, we improved the generation performance with respect to other traffic generators. In this paper we present only the transport layer generation issue, aiming at analyzing ITG rate performance. The final section reports a picture (Figure 5) comparing the performance of our ITG to those of the other presented traffic generators.

III. Internet Traffic Generator

Internet Traffic Generator (ITG) currently runs under Linux and FreeBSD operating systems. In this work we present the details related to the transport layer traffic generation. ITG is made of two instruments: One-way-delay Meter (OWDM) and Round-trip-time Meter (RTTM), each of which is constituted by two utilities, a sender and a receiver.

Both packet inter-departure and packet size are modeled as independent and identically distributed (i.i.d.) series of random variables. Currently, several choices are available for these random variables: constant, uniform, normal, cauchy, pareto and exponential are some of the implemented distributions. The user can choose one of these distributions for inter-departure and packet size random variables in order to generate the desired traffic pattern. However, thanks to the Robert Davies' random number generator library [18], it is very simple to add new random distributions, so as to simulate a rich variety of traffic sources. As said before, ITG allows to send traffic according to the theoretical models for various protocols. This means that the user can simply choose one of the implemented protocols. The distributions and the corresponding parameters for the inter-departure and packet size random variables are automatically determined by ITG. Another feature of ITG is the possibility of specifying the seed value for the packet inter-departure and packet size random processes; in this way, it is possible to repeat exactly a particular realization of these random processes. This provides for the reproducibility of experiments. To collect statistics it is necessary to store some information in the transmitted packets. We store the number of the flow the packet belongs to, a sequence number and the time the packet was sent in the transport layer payload. The receiver computes the transmission time and the throughput based on this information and the measurement of the time in which receives the packets. The sender can also log in a file the information included in the packets it sends. This allows us to retrieve information not only about received traffic but also about generated traffic.

Finally, we want to point out that ITG is not in charge of the clock synchronization between sender and receiver, required in one-way delay measurements. This synchronization is demanded to appropriate protocols.

A. Software architecture

OWDM and RTTM are written in C++ language. In case multiple flows are to be generated, child processes are created to handle each flow. This allowed us to write the remaining code as we had only one flow, and this obviously made data structures and code simpler. Then each process parses its own arguments and recognized tokens are analyzed by the option parser, which checks for their correctness and sets the appropriate variables. On success, a socket is opened, whose type depends on the transport protocol type. In order to speed up the logging operation, ITG uses a buffer to temporarily store the log information. When the buffer is full, its content is stored into a binary log file. This strategy enables to reduce the access rate to the hard disk and the interference on the generation and reception tasks.

In the traffic generation performance task it is important to take into account CPU scheduling: several processes (both user and kernel level) can be running on the same PC and this has a bad impact on the generation performance. Since ITG currently runs under non real-time operating systems, the support of real-time applications is not very efficient (due to their scheduling mechanism and the inevitable timer granularity). It was therefore necessary to use a trick. A variable records the time elapsed since the last packet was sent; when the inter-departure time must be awaited, this variable is updated. If its value is less than inter-departure time the remaining time is awaited, otherwise the inter-departure time is subtracted from the value of this variable and no time is awaited. This strategy guarantees the required bit rate, even in presence of a non real-time operating system. Disregarding this aspect would result in very poor performance that is the generated traffic would be much lower than requested traffic. Another property of our generator in the performance field is the possibility of setting an high priority for the generation process (this feature is available in RUDE/CRUDE generator too). If supported by the operating system, this feature allows to achieve even better performance.

IV. Transport Layer Traffic Generation

Now we show several examples of how to use our tool in the one-way-delay mode (OWDM) and what kind of information can be obtained by analyzing log files. Since the log files of ITG have the same format of those produced by MGEN, we used MGEN's utilities (*malc* and *ez*) to obtain the results shown in this section. The target of these examples is to show the behavior in the generation phase and to ver-

ify that the traffic requirements are met. ITG implements both TCP and UDP traffic generation according to several statistical distributions. In this section we show some combinations of IDT (Inter Departure Time) and PS (Packet Size) random variables. We present two generation trials, one TCP traffic generation and one UDP traffic generation. The two PCs used for these trials are named "PCSender" (sender) and "PCReceiver" (receiver). The relevant hardware and software details are the following:

- PCReceiver: Intel Celeron 400 Mhz, 64 Mb RAM and 128 Kb cache memory, Linux Red Hat 7.1 - kernel 2.4.2-2;
- PCSender: Intel PII 850 Mhz, 128 Mb RAM and 256 Kb cache memory, Linux Red Hat 7.1 - kernel 2.4.2-2.

OWDM sender logs sent packet in a file that we process using *ez*, in order to obtain the bit rate graph of generated flows. Finally we underline that in the following examples we generate only one flow for each trial. ITG is capable to generate multiple flows: for example, in the network analysis, it is possible to combine many flows related to several applications (HTTP, VoIP, FTP,...).

TCP, IDT = Pareto distributed and PS = constant

We use the following command on the PCSender:

```
./ITGsend -a PCReceiver -l send -t 60000
-T TCP -V 3 10 -c 16
```

This command tells the sender to generate one flow addressed to the host named PCReceiver with TCP transport protocol.

The payload size of all packets is constant and equal to 16 bytes. The flow lasts 60 seconds and the inter-departure is a Pareto random variable, characterized by shape equal to 3 and scale equal to 10. We want to calculate now the expected average bit rate, in order to verify the accuracy of our ITG. First, we note that *mcalc* and *ez* utilities consider only the payload size of packets (and not their full size) in determining the average bit rate, and we will do so, too. The mean of a Pareto random variable having the above defined parameters is equal to 15, this means that the average time interval between the departure of two consecutive packets is 15 msecs; the payload size is 16 bytes (=128 bits). Therefore the average bit rate is equal to the ratio between 128 bits and 15 msecs, which yields 8,533Kbps. The *mcalc* output and resulting plots derived from the log file of PCsender are shown in Figure 1 and Figure 2 and confirm our expectations. Note that the bit rate plot (Figure 2) needs the specification of a window size that is the time interval on which the bit rate must be computed. Moreover, please note that all the parameters that you can see from the mcalc output are related to the generated traffic, even though they are addressed as "received" (this is due to the fact that MGEN doesn't log sent packets and therefore analyzes only receiver's log files).

```
Num pkts recvd  :  4010
Join delay      : 73979.945 sec
Recv pkt rate   :    66.939 pkt/sec
Recv data rate  :     8.570 kbps
Pkts dropped    :     0
Ave. Tx Delay   :     0.000 sec
Max. Tx Delay   :     0.000 sec
Min. Tx Delay   :     0.000 sec
Delay variation :     0.000 sec
```

Fig. 1. TCP traffic generation: statistics

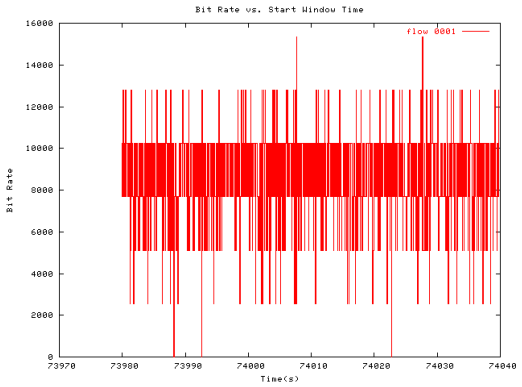

Fig. 2. TCP traffic generation: bit rate plot

UDP, IDT = Exponentially distributed and PS = uniformly distributed

We use the following command on the PCSender:

```
./ITGsend -a PCReceiver -l send -t 60000
-T UDP -E 2 -u 1000 1450
```

This command tells the sender to generate one flow addressed to the host named PCReceiver with UDP transport protocol.

```
Num pkts recvd  :   120
Join delay      : 42303.020 sec
Recv pkt rate   :     2.220 pkt/sec
Recv data rate  :    21.808 kbps
Pkts dropped    :     0
Ave. Tx Delay   :     0.000 sec
Max. Tx Delay   :     0.000 sec
Min. Tx Delay   :     0.000 sec
Delay variation :     0.000 sec
```

Fig. 3. UDP traffic generation: statistics

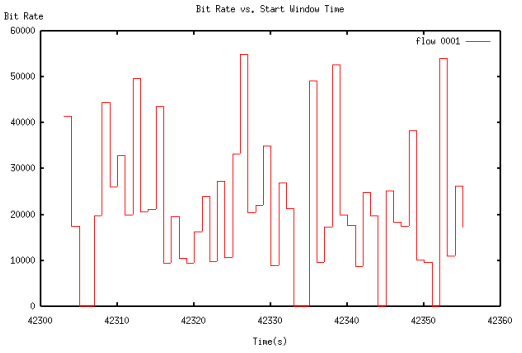

Fig. 4. UDP traffic generation: bit rate plot

The payload size of all packets is an uniformly distributed random variable between min=1000 bytes

and max=1450 bytes, that is, considering Ethernet, IP and UDP headers, their full size is 1042 and 1492 bytes respectively. The flow lasts 60 seconds and packet generation process is an exponentially distributed random variable with mean = 0,5 . The average bit rate is 19,6kpbs . The output results and the bit rate plot of this trial are shown in Figure 3 and Figure 4.

V. Conclusion and future work

In this work we presented a general framework for traffic models and our Internet Traffic Generation tool. ITG has been planned for generating network traffic (ICMP), transport layer traffic (TCP and UDP) and "layer 5-7" traffic (HTTP, FTP, TELNET, SMTP, DNS, VoIP, Video, NNTP, ...). In this paper we presented only the UDP and TCP generation. ITG implements both TCP and UDP traffic generation according to several statistical distributions (exponential, uniform, constant, pareto, cauchy, normal,...) both for IDT (Inter Departure Times) and PS (Packet Size) random variables. ITG enables to simulate various network conditions under different network traffic loads and network configurations. ITG is based on theoretical traffic models and represents a way for analyzing network performance through the measurement of the one-way-delay and the round-trip-time of packets that traverse the network. ITG steps from another our tool named Mtools [13], [14]. The basic idea of creating a new traffic generator arose from the lacks of existing ones (MGEN, Rude/Crude, etc.), emerged when we used them to analyze the different behavior of the network when some strategies that provide QoS are employed. The features we needed were:

- the possibility of simulating more traffic sources, repeating many times exactly the same traffic pattern (not only its mean values) and getting information not only about received packets but also about transmitted packets
- the possibility to generate both UDP and TCP traffic and several "layer 5-7" traffic (Http, Ftp, Telnet, Smtp, Dns, VoIP, Video, Nntp,...)
- the increase of both the generated and received data rate with respect to the other traffic generators. Figure 5 illustrates the "Expected behavior" compared with several real data rate (the packet size is fixed and equal to 1024 bytes). It is possible to note that ITG performs better than Rude/Crude, TG 2002 and MGEN. We used the testbed shown in Figure 6 to carry out the rate performance tests
- the possibility to test new network scenario such as "heterogeneous networks". We are experimenting an heterogeneous scenario made of both heterogeneous end user devices (PC desktop, Laptop, PDA) and heterogeneous network technologies (Ethernet wired, Wireless LAN 802.11b and 802.11g, GPRS). We are using ITG for network characterization and interoperability measurement

For these reasons we decided to build a traffic generator which satisfies these requirements and also includes a tool for measuring the round trip time. ITG

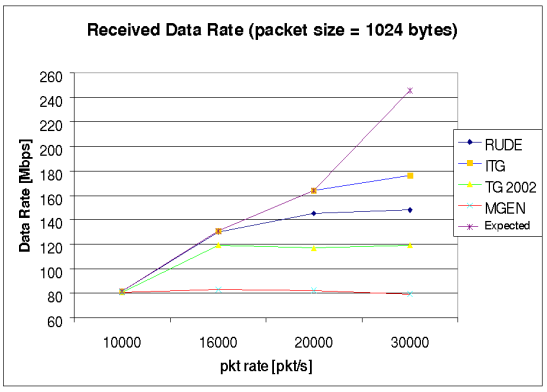

Fig. 5. Data rate analysis

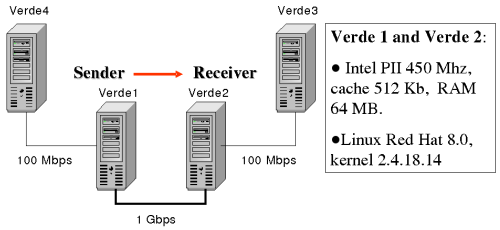

Fig. 6. Testbed for data rate analysis

is easily extensible to support other features (now we are working on porting to Win32 operating systems too). As far as future work, by comparing ITG to the other traffic generators we are working on performance issues related to the generated packet rate. The main next step is the full implementation of all presented "layer 7" traffic and a meticulous performance analysis both on our experimental testbed and our real academic network architecture.

Another future work is the implementation of new tools, to be included in our ITG, for the presentation of the results of the trials: in this first stage of our work we are using *mcalc* for the output statistics and *ez* for the graphics, but for the next future we are planning to build our own utilities.

ITG is currently downloadable and freely available at `www.grid.unina.it/software/ITG`. ITG has been used from other research groups in order to do real testing on their networks [19], [20].

Currently we are working on a "mobile version" of our ITG. We are working on porting ITG under PDA platform with Linux FAMILIAR (kernel 2.4.18) operating system. This implementation would make it possible to carry out a complete characterization of a real heterogeneous mobile network [21]. After this step we are planning to use ITG (with mobile extension) in a wide range of access networks. Mobile Internet access using WLAN and GPRS/3G has gained good popularity. We would test ITG in a more large heterogeneous environment made by both heterogeneous (wired and wireless) network (WLAN, Bluetooth, UMTS, GPRS, GSM,...) and heterogeneous users' device (Laptop,

PDA, Advanced Mobile Phone, Workstation,. . .) [22].

The results carried out thanks to ITG can be used as a reference scenario for complete characterization of heterogeneous networks and for development of communication applications over heterogeneous networks.

Acknowledgments

This work has been carried out partially under the financial support of the "Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR)" in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)".

References

[1] M. Zukerman, T.D. Neame and R.G. Addie, *Internet Traffic Modeling and Future Technology Implications*, Proceedings of Infocom 2003

[2] URL http://www.fokus.gmd.de/research/cc/glone/employees/sebastian.zander/private/trafficgen.html

[3] URL http://www.caip.rutgers.edu/ arni/linux/tg1.html

[4] URL http://www.ittc.ku.edu/netspec/

[5] URL http://www.netperf.org/

[6] URL http://playground.sun.com/psh/

[7] URL http://manimac.itd.nrl.navy.mil/MGEN/

[8] URL http://www.atm.tut.fi/rude

[9] http://www.fokus.fhg.de/usr/sebastian.zander/private/udpgen

[10] D. Papaleo, S. Salsano, *The Linux Traffic Generator*, 3rd ed. INFOCOM Department Report 003-004-1999 - University of Rome "La Sapienza"

[11] IETF IP performance Metric (IPPM) Working Group - http://www.ietf.org/html.charters/ippmcharter

[12] P.E. McKenney, D.Y. Lee, B.A. Denny, *Traffic Generator Software* Release Notes, SRI International and USC/ISI Postel Center for Experimental - January 8, 2002

[13] S. Avallone, A. Pescapè, S.P. Romano, M. Esposito, G. Ventre, *Mtools: a one-way-delay and round-trip-time meter* 6th WSEAS International Conference, ISBN 960-8052-63-7

[14] S. Avallone, A. Pescapè, M. D'Arienzo, S.P. Romano, M. Esposito, G. Ventre, *Mtools* IEEE Network, Software Tools for Networking - September/October 2002, Vol. 16 No. 5 pag. 3. ISSN 0890-8044

[15] URL http://rsandila.ezfish.net/traffic.html

[16] URL http://pacgen.sourceforge.net/

[17] URL http://tochna.technion.ac.il/project/NTGen/html/ntgen.htm

[18] R. Davies, *Newran02A - a random number generator library* URL http://webnz.com/robert/nr02doc.htm

[19] P. Salvo Rossi, G. Romano, F. Palmieri,G. Iannello, *Bayesian Modelling for Packet Channels* XIV Italian Workshop on Neural Nets, WIRN Vietri 2003

[20] A. Pescapè, M. D'Arienzo, G. Ventre, *Transparent and Automatic SLA Management: moving towards Proactive Networks* to appear in the proceedings of the International Conference on Networking 2004

[21] A. Pescapè, S. Avallone, G. Ventre, *Distributed Internet Traffic Generator (D-ITG): analysis and experimentation over heterogeneous networks*, accepted poster at ICNP 2003 (http://icnp03.cc.gatech.edu/)

[22] G. Iannello, A. Pescapè, G. Ventre, L. Vollero, *Experimental analysis of heterogeneous wireless networks* accepted at WWIC 2004 (www.wwic2004.de)