UNIVERSITY CARLOS III of MADRID

Department of Telematics Engineering

Master of Science Thesis
in Telematics Engineering

Experimental Assessment of Traffic Generators

Author: **Georgios Z. Papadopoulos**
B.Sc. in Computer Engineering

Supervisor: **Pablo Serrano Yáñez-Mingot, Ph.D.**

Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés - Madrid - España

July 2012

M.Sc. Thesis in Telematics Engineering

Experimental Assessment of Traffic Generators

Author: **Georgios Z. Papadopoulos**
B.Sc. in Computer Engineering

Αφιερωμένο στους παπούδες μου, Βενιαμίν - Μαρία και Παύλος - Ανατολία

Abstract

There exist plenty of internet traffic generators aimed at generating and transmitting packets, and measuring the performance of computer networks. Hence, the selection of the proper tool becomes a hard task, and evaluating which one fulfills best the prospect task is very time-consuming. In this work we assess the performance of four representative (and most popular) network packet generators in a laboratory environment in different conditions and under a wide set of experiments. The results demonstrate that the different nature of the analyzed tools provide different performance under different conditions. Thus, the main goal of this work is to provide researchers and IT administrators with an important set of performance results under real-life constraints.

Acknowledgements

The current thesis is the result of the past months of intensive and exciting work at labs of University Carlos III of Madrid, Spain where I got a flavor of conducting research and about the research process cycle. I would like to gratefully and sincerely thank my supervisor Dr. Pablo Serrano, Associate Professor in the Department of Telematics at UC3M, for his guidance, understanding, patience, constant support and most importantly, his friendship during this period. His mentorship was paramount in providing a well rounded experience consistent my long-term career goals.

Special thanks go also to my thesis advisor Andrés García Saavedra, Ph.D. Candidate in Telematics Engineering at UC3M, for his constant technical support, for hints during the implementation process, for providing uncountable wise advices and for his collaboration in research activity.

I deeply thank Dr. Albert Banch and Dr. Periklis Chatzimisios for their confidence regarding to my skills to pursue my M.Sc. studies and thesis at UC3M.

I want to thank as well my flatmate, all my friends and colleagues (lab 4.1 A03), which made this period of my life an experience that I will never forget.

Finally, this thesis would not have been possible without the help of three persons. I thank my parents, Zouramp and Larisa and my sister, Maria, for their wholehearted support and believe in me throughout my entire life.

Georgios Z. Papadopoulos

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As computer networks have become increasingly ubiquitous the more researchers are increasingly focused on optimizing and improving the performance of computer networks in various statistical metrics like throughput, delay, jitter or packet loss. Network traffic generators (TG), applications that generate and transmit packets over the network, are used by researchers and IT administrators in order to analyze the performance of their networks: configuration, algorithms or protocols evaluation.

Nowadays plenty of TGs are available (see Table 1.1) for the purpose described above. However, not all of them offer the same features nor provide the same performance under different conditions. Our experimental results indicate that different TGs have very different usage of the HW resources, and provide different results when dealing with short packet inter-departure times (i.e. high packet rate generation).

These differences in the TGs behavior can have a non-negligible impact as users might be running into wrong conclusions assuming ideal packet generation. Furthermore, a thorough evaluation of all the available tools to generate network traffic adds a prohibitive delay in the testbed deployment. In order to illustrate this problem, let us refer to Gamess and Velásquez [13]: They had to use two different performance benchmarks and to write their own tool in order to evaluate the IPv4 and IPv6 forwarding performance.

The primary contribution of this work is to classify the TGs according to their characteristics and features (i.e. TGs that provide information about "Delay or Jitter"), and to investigate their performance by using different metrics and testbed configurations. The goal of this work is to provide researchers and IT engineers an answer regarding the TG tool they should use according to the planned experiment.

The rest of the thesis is organized as follows. Chapter I, contains a detailed description of the selected traffic generators and previous works related to our study are presented. In Chapter II, the test-ted and methodology that we followed in this research is discussed. Chapter III, the performance assessment in terms of throughput, delay and CPU consumption is presented. In Chapter IV, the sensitivity analysis of resource shortage is presented. Conclusion and future work are discussed in Chapter V.

1.2 The Selected Traffic Generators

After a thorough research on the web [11], [27], [17] and [28] of the state of art of existing TGs, we gathered an exhaustive list of network traffic generators which is presented in Table 1.1. From this pool of TGs, we have selected four different network traffic generators for our study: Iperf, D-ITG, MGEN and RUDE & CRUDE. The reasons that led us towards this choice are mainly the following. **Popularity**. Figure 1.1 shows a classification of TGs according to the number of citations in research articles and patents. **Availability**. The selected TGs are easily available online at a very low cost (e.g. LANforge [10], used in [31], has been discarded due to its high price which ranges from $1999 (USD)). **Activity**. Being an active development project was a constraint in our selection. Moreover, we kept track of their popularity along the last years and filtered out those with descending usage trends (see Figure 1.2). Note that RUDE & CRUDE has a significant number of citations even though the last release was in September 2002. **Nature**. Those TGs designed for a specific purpose has also been discarded, e.g. "PacGen" for is kernel based generator, "packETH" is a tool that explicitly aimed at ethernet links [27]), "TCPivo" requires several patches to the Linux kernel [27]).

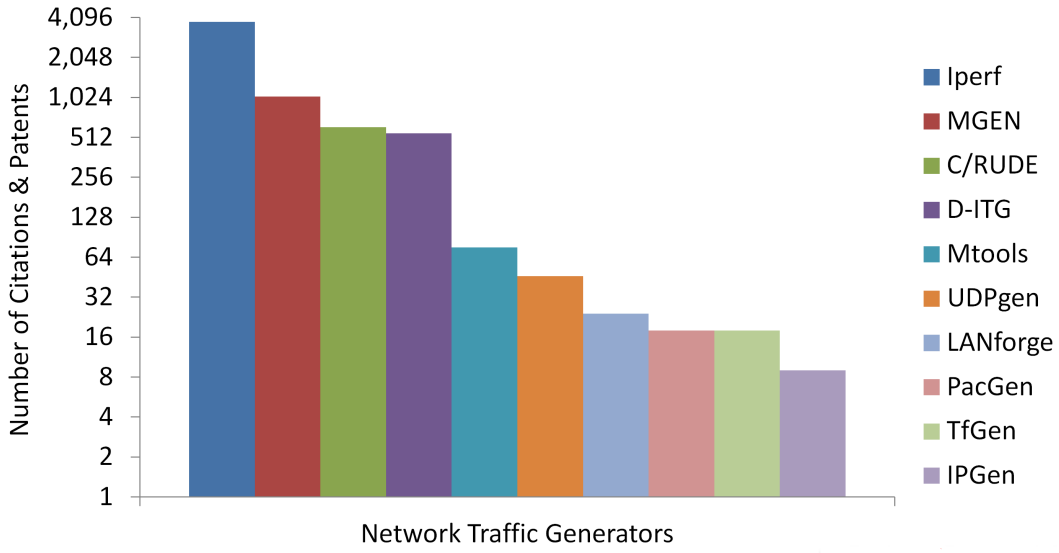

Figure 1.1: Popularity of Traffic Generators

The survey evaluating the current popularity of TGs (figure 1.1), and the study of the activity trends along the time (figure 1.2) took place in July 03, 2012 using the well known Google Scholar and IEEE Xplore search tools [19], [16]. Note that, even being methodologist in the procedure, the results cannot be considered as perfectly accurate, however they still serve us as a coarse monitor of the current trends.

According to figure 1.1, Iperf is the most popular TG with a fairly big difference with respect to the second one (MGEN) and the rest. What's more, according to figure 1.2, Iperf's trend of popularity grows as a much higher rate than the others.

Table 1.1: List of Traffic Generators

Free Tools		
	Mtools	MGEN
	D-ITG	Iperf
	TrafGen	Harpoon
	Netperf	GEIST
	Ostinato	Trafgen
	IP-Packet	GenSyn
	Packet Shell	TfGen
	UDP Generator	PacGen
	TTCP, Test TCP	PIM-SM
	RUDE & CRUDE	UDPgen
	Traffic Generator Tool	NTGen
	FTP traffic generator	Surge
	SPAK, Packet Generator	Netspec
	Poisson Traffic Generator	MxTraf
	Network Traffic Generator	KUTE
	Self Similar Traffic Generator	IPGen
	Jugi's Traffic Generator (jtg)	TCPivo
	Real-Time Voice Traffic Generator	packETH
	epb - Ethernet Package Bombardier	
	Brawny and Rough Udp Traffic Engine	
Commercial Tools		
	SiteSpy	
	Bit-Twist	
	ByteBlower	
	ProvaGEN 3.0	
	Candela Technologies LANforge	
	LANdecoder32T Traffic Generator	
	6WINDGate fast path traffic generator	
	Omnicor Software IP traffic generators	
	Internetworking Test Traffic Generation	
	Omnicor Hardware IP traffic generators	
	Skaion's Traffic Generation System (TGS)	
	Traffic Generator for Wide Area Networks	
	LANTraffic V2 and IP Traffic Test & Measure	

1.2.1 Iperf

Iperf is an open source tool, was developed by the Distributed Applications Support Team (DAST) at the National Laboratory for Applied Network Research (NLANR). The newest version of Iperf, version 1.7, is used for the evaluation of both TCP and UDP traffic performance with both unicast and multicast transmission. Iperf reports various statistical measurements like throughput, delay jitter and

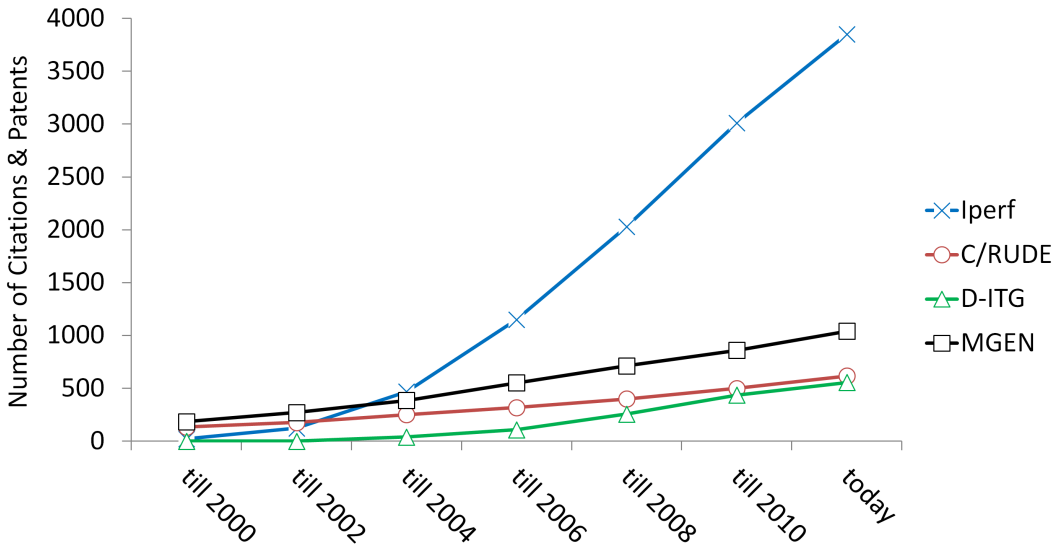

Figure 1.2: The Lifetime of Traffic Generators

datagram loss. Typical Iperf output contains a timestamped report of the amount of data transferred. Although, Iperf is a command line performance tool, there is a GUI interface version of Iperf, named Jperf which was developed in Java. Iperf allows the user to tune several UDP/TCP parameters that can be used for testing a network. Furthermore, Iperf handles multiple parallel transfers. Iperf is open source software and is currently available on Linux, Unix and Windows platforms with the same command options. Finally, Iperf follows the client-server model and is designed to work with both IPv4 and IPv6 [5].

1.2.2 D-ITG

Distributed Internet Traffic Generator (D-ITG) is an open source traffic generator was developed at the University of Napoli "Federico II", in the Department of Informatics and Systematic. Currently available on Linux, Windows, OSX and Linux Familiar platforms with both command line and graphic user interface. Capable of producing traffic at packet level according to predefined variables, inter-departure time and packet size. D-ITG provides a variety of probability distributions like: Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson and Gamma. Furthermore, it supports both IPv4 and IPv6 traffic generation and can generate traffic at network (e.g. ICMP, IPv4), transport (e.g. TCP and UDP), and application layer (i.e. DNS, Telnet and VoIP). D-ITG provides metrics for both one-way-delay (OWD) and round-trip-time (RTT) measurement, packet loss evaluation, jitter and throughput measurement. Finally, D-ITG permits the setting of TOS (DS) and TTL packet fields [9].

D-ITG platform consists of four distributed components: *ITGSend*, *ITGRecv*, *ITGLog*, and *ITGDec*. Figure 1.3 shows a graphical overview of the relationship among the components. A short description of the four main components of D-ITG

architecture follows.

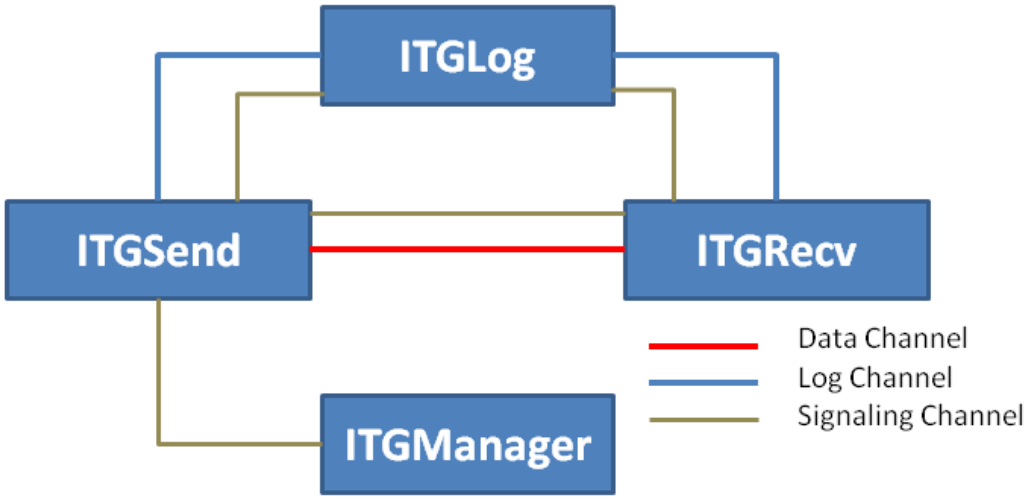

Figure 1.3: D-ITG Software Architecture [9]

D-ITG follows the client-server model. ITGSend is the sender component (the client) which can operate in different modes: can generate either single flow mode or multiple data flows simultaneously as specified by the input file (configuration file). ITGRecv acts as the server in order to receive the data flows. Both generate log files with flow information which can be stored locally or remotely using ITGLog which is the log server for the platform. Finally, with use of ITGDec is possible to analyze the results of the experiments.

1.2.3 MGEN

The Multi-Generator (MGEN) is an open source software developed by the research group named PROTocol Engineering Advanced Networking (PROTEAN) at the Naval Research Laboratory (NRL). MGEN supports both IPv4 and IPv6 traffic generation at network level, and it works with UDP and TCP traffic at transport layer. MGEN follows the client-server model and is capable to transmit in unicast, multicast and in broadcast traffic generation. The generated traffic can be received and logged for analyses by using other tools (e.g. TRPR [4]). Hence, by analyzing the log file with TRPR the performance statistics on throughput, delay and so on can be calculated.

Number of traffic streams can be generated by MGEN (i.e. Periodic, Poisson, Burst, Jitter and Clone), the periodic has the same philosophy of constant while the clone pattern incrementally read a file of the specified “fileType” to determine packet sizes and transmission intervals, currently only tcpdump binary files are supported. For all traffic profiles the size and the rate of packets can be scheduled to be changed at any time during the flow generation. Finally, it is possible to have more than one

stream simultaneously as specified by the input file.

Currently, MGEN runs on Unix-based (including MacOS X) and Windows platforms. Moreover, it can be used in network simulation environments (e.g. ns-2 and Opnet). The latest version of MGEN is 5.0, previous versions (MGEN 4.x and MGEN 3.x) are available, but they are not interoperable while only MGEN 3.x has a graphical user interface [3].

1.2.4 RUDE & CRUDE

RUDE & CRUDE (C/RUDE), Real-time UDP Data Emitter (RUDE) and Collector for RUDE (CRUDE), are both command line open source programs and both programs were developed at the Tampere University of Technology as part of the Faster 2000 project. RUDE (client) generates and transmits packets to the network according to the specification in its configuration file while CRUDE (server) receives and logs the generated traffic by RUDE, always following the client-server model.

Currently, C/RUDE can generate and measure only UDP traffic [12], supports IPv4, and is capable to run over Linux, Solaris and FreeBSD platforms. Two types of traffic streams can be generated, the CONSTANT where traffic consists of a constant flow of UDP packets and TRACE stream consists of UDP packets with packet sizes and inter-packet gaps specified separately for each packet in a trace file. In MGEN and C/RUDE in both traffic profiles the size and the rate of packets can be scheduled to be changed at any time during the flow generation and it can initiate multiple CONSTANT or TRACE streams simultaneously.

C/RUDE's operation and configuration is similar to the traffic generator that was described previously (i.e. MGEN). C/RUDE was designed to overcome an accuracy limitation present in MGEN, due to the use of system timers (e.g. Linux kernel) which is limited the timer resolution to 10ms which is quite poor value. Hence, RUDE can dispatch UDP packets according to a predefined pattern with steady precision of 1 microsecond (provided that the PC's clock are precisely adjusted). In order to achieve this precision, CRUDE generates a snapshot about each received packet. The snapshot includes the following information: the destination address, a stream ID specified in RUDE configuration, sequence number of the packet within the stream, timestamp with 1 microsecond of resolution indicating when each packet has been sent, receiving timestamp and the packet size in bytes. The snapshots can be displayed either on-the-fly in a text form on the standard output or logged in a binary form to the file to be decoded into the text form later where the last option reduces I/O operations during measurement allowing for packet processing at higher speeds. C/RUDE has also lower and stable overhead, thus, additional delay is lower. Finally, provide plenty of statistical measurements information (i.e. throughput, delay, jitter, packet loss and inter-arrival).

1.3 Comparison of Traffic Generators

In this subsection, a first look comparison among the TGs will be presented by providing the pros and cons of each TGs described above. After a deep study of

Table 1.2: The Features of Traffic Generators

Features	Iperf	D-ITG	MGEN	C/RUDE
First Release	02/2000	2003	1993-1995	01/2000
Last Release	07/2010	08/2011	05/2011	09/2002
Last Version	2.0.5	2.8.0-rc1	5.02	0.70
Privileges	User	User	User	Root
Interface	Console, GUI	Console, GUI	Console, GUI	Console
Log File	No	Yes	Yes	Yes
Suported Platforms	Linux, Windows	Linux, Windows	Linux, Windows	Linux
Network Protocols	IPv4, IPv6	IPv4, IPv6, ICMPv4 ICMPv6	IPv4, IPv6	IPv4
Transport Protocols	TCP, UDP	TCP, UDP, SCTP DCCP	TCP, UDP	UDP
Application Layer	-	DNS, Telnet VoIP	-	-
Multicast	Yes	No	Yes	No
Broadcast	No	No	Yes	No
Traffic Profiles	Constant	Constant, Pareto, Uniformly, Exponentially, Cauchy, Normal, Poisson, Gamma, Weibull & Burst	Periodic, Poisson, Burst, Jitter, & Clone	Constant & Trace

every TGs' features we conclude that, a priori, there isn't a TG that is better suited for all tasks, since according to their characteristics each TG has its own advantages and disadvantages.

Popularity and activity First, as depicted in figure 1.2, we observe that Iperf's popularity is increasing at a higher rate with the time with respect to the rest. D-ITG, even though is the most recent tool, it is currently as active as C/RUDE and MGEN. Finally, it is worth mentioning that, even though the last release of C/RUDE was about ten years ago, it still holds an ascending trend.

Protocols and platform Excluding C/RUDE, all TGs support both TCP and UDP protocols, they are capable of running on Windows platforms (not only Linux) and support IPv6. D-ITG, differently from the rest of TGs, provides support for a wide set of other protocols such as SCTP or DCCP in the transport layer, ICMPv4 and ICMPv6 in network layer and also protocols for application layer like Telnet, DNS and VoIP. Moreover, MGEN is capable to transmit packets in multicast and broadcast mode in addition to unicast while among the rest, only Iperf supports multicast.

Traffic patterns Iperf has a significant drawback since it only supports Constant Bit Rate (CBR) while C/RUDE supports an additional traffic profile called TRACE, which is customized by the user. MGEN supports CBR and four traffic profiles more: Poisson, Burst, Jitter and Clone (similar to TRACE in C/RUDE). D-ITG, in turn, supports more traffic patterns than any other (i.e. Uniformly, Exponentially, Pareto, Cauchy, Normal, Poisson, Gamma, Weibull and Burst).

Statistics report C/RUDE is the one that provides a more complete report of metrics (i.e. Throughput, Delay, Jitter, Packet Loss and Interarrival statistics). Tables 1.3 and 1.2 provide a closer overview of the figures reported by each traffic generator.

Usability Iperf from our experience is the easiest tool to install and to learn how to use it. The rest three TGs use an input file that contains all the necessary configured parameters in order to run the experiment. Hence, these TGs are more flexible compared with traditional command line arguments. Note that, D-ITG supports command line argument as well. Furthermore, even though D-ITG and MGEN come with complete and detailed documentation, it is hard to learn how to use it due to the plenty of options that they support. Finally, with MGEN in order to obtain the statistical results of an experiment there is a need to use another program (e.g. TRPR) which makes it even more complex and resulting in an additional time for learning the new tool.

Table 1.3: The Statistical Measurements of TGs

Metrics	Iperf	D-ITG	MGEN	C/RUDE
Throughput	Yes	Yes	Yes	Yes
Delay	No	Yes	Yes	Yes
Jitter	Yes	Yes	No	Yes
Packet Loss	Yes	Yes	Yes	Yes
Interarrival	No	No	Yes	Yes

1.4 State of the Art

In [8] and [7] Avallone et al., the authors of D-ITG TG, present an exhaustive description about the architecture of their tool, and an experimental evaluation of the same compared against other popular traffic generators: Mtools [6], RUDE & CRUDE [12], MGEN [3], Iperf [5], UDP Generator [26] and TG2002 [20]). They mainly focus on analyzing the throughput performance, the receiving and generating data rate, of a testbed where two PCs (i.e. with operation systems either Windows or Linux, or combination of two) were connected back-to-back via a gigabit ethernet link and using UDP as transport protocol. According to their results, they claim that D-ITG performs better regarding to the throughput comparing with the rest TGs and is fairly close to the expected values. In this study, the authors provide results only about the throughput, their experiments took place for various packet rates but always having the same packet size (i.e. 1024 Bytes).

Kolahi et al. [22] present a performance comparison of four network TGs: Iperf [5], Netperf [21], D-ITG [9] and IP Traffic [29] in a laboratory environment. Their experimental testbed consists of two computers with Windows platform installed, and connected via Ethernet of 100 Mbps link end to end. They performed experiments for various payload sizes, ranging from 128 Bytes to 1408 Bytes and using TCP as transport protocol. Their results show that the TGs can produce significantly different results for the same network set up. In particular Iperf provided the highest throughput (93.1 Mbps) while IP Traffic the lowest (76.7 Mbps). However, in cases of low packet sizes, 128 Bytes and 256 Bytes, IP Txraffic performed much better (i.e. 61 Mbps) while D-ITG not so well (i.e. 38.1 Mbps). While this results already indicate the high impact of choosing the correct generator tool, their experiments focused specifically on Windows platform and on TCP transport protocol, obtaining results again only for throughput performance.

Karima et al. [31] give a detailed description and analysis of eight network benchmarking tools (i.e. Netperf, D-ITG, NetStress [25], MGEN, LANforge [10], Network Traffic Generator [15], RUDE & CRUDE, and WlanTV [30]). They only provide a view of their main features, i.e., assessment was classified in installation, testing, usability, supported platforms and protocols. They conclude that there is not a real winner among the TGs that they analyzed and that the researchers must work with several TGs in order to perform basic experiments. However they did not run any real-life testbed and, thus, the conclusion is merely an abstract discussion.

To the best of our knowledge, current literature still lacks of accurate and thorough experimental evaluation of the heterogeneity of network packet generators. This heterogeneity and its impact have already been detected in existing works, but we still miss a valid assessment on their performance. Hence, as a first step in our study, we present a deep look at the features and usability of each analyzed traffic generator. Then we perform number of set of experiments under different situations and looking at the results from different perspectives to obtain a better understanding on their performance.

Chapter 2

Test-bed & Methodology

2.1 Test-bed

The experimental work of this master thesis was carried out using two desktop PCs using Linux Ubuntu with Kernel 3.0.0 as OS platform. The testbed consists of two terminals that have been connected within the same subnet using a switch (i.e. D-Link DES-1008D) and 100Mbps Fast Ethernet network interfaces. Figure 2.1 depicts this simple topology. The PCs used for the experiments are two powerful Intel(R) Core(TM)2 Quad CPU Q 8400 at 2.66GHz CPU, with 4096 MB of RAM memory, and a Seagate with Spin Speed of 7200 rpm and capacity of 1TB. See Table 2.1 for a comprehensive summary. These machines allowed us for controlled experiments assuming that there are not external HW/SW limitations.

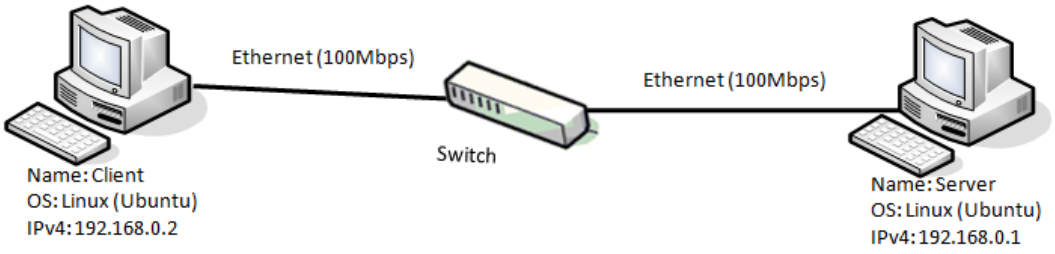

Figure 2.1: Experimental Topology in Wired Connection

2.2 Methodology

The focus of this work is the evaluation of the actual generation of existing traffic generators. For this reason, all experiments have been run using UDP traffic to avoid any automatism in the congestion management and flow generation.

Along the experiments, we created a wide set of different traffic configurations (i.e. packet size, packet rate) and analyzed the results obtained under different conditions (i.e. available HW resources). All experiments have been executed for 20 seconds

Table 2.1: Experiment's Setup

CPU	Intel(R) Core(TM)2 Quad CPU Q 8400 @ 2.66GHz
RAM	4096 MB
Hard Disk	Seagate Barracuda, 7200 series & Capacity of 1TB
Network Card	Intel (R) 82567 Gigabit Ethernet Controller
Wired	Ethernet Crossover Cable at 100 Mbps
Wireless	IEEE 802.11a with MCS at 6 and 54 Mbps
Operating System	Linux Ubuntu 11.10, Kernel 3.0.0-19-generic
Duration	20 Seconds
Transport Protocol	UDP
Traffic Profile	Constant Bit Rate

with a sufficient number of iterations to obtain reasonable confidence intervals of the statistics shown.

It is worth noting that different generators provide different configuration interfaces and input parameters. For example, Iperf only allows setting the sending bitrate (i.e. instead of sending packet rate). All experiments have been run using all traffic generators, setting the corresponding parameters to obtain the same traffic generation configuration: traffic pattern, packet rate and packet size.

2.2.1 Throughput

Throughput is the first metric analyzed, that is, data bitrate successfully received by the server terminal. Most of the traffic generators (e.g. iperf) provide this information on-the-fly or as a resulting report. Other tools (e.g. MGEN, D-ITG, C/RUDE) allows for logging all the incoming packets with enough information for post-processing. TRPR [4], ITGDec [9], and a provided *Perl* [1] script, respectively, were used for generating statistical reports.

2.2.2 Delay

In order to avoid clock synchronization on the terminals which may has crucial impact on the delay and knowing that the "One Way Delay" (OWD) on Fast Ethernet is "fixed" due to the nature of the wired link, thus, the goal of this work is to evaluate the performance of TGs according to their operational accuracy in terms of the Inter-Departure Time (IDT) process. In other words, whether the TGs do transmit packets constantly every fixed and determined time when using CBR traffic (see Figure 2.2). For this evaluation, we used *TShark* application [32], at the client side, in order to sniff and check the timestamps of the packets being transmitted. Afterwards, by using the empirical Cumulative Distribution Function (CDF) in Matlab we plotted these data. Thus, in case there is a non expected performance on the CDF graph it means that there would be respectively an impact on the delay performance as well since as we mentioned previously the OWD is fixed in ethernet link.

The CDF provide us a qualitative overview, variance, of the IDT performance; however this information is not enough for having the whole picture, thus, our next

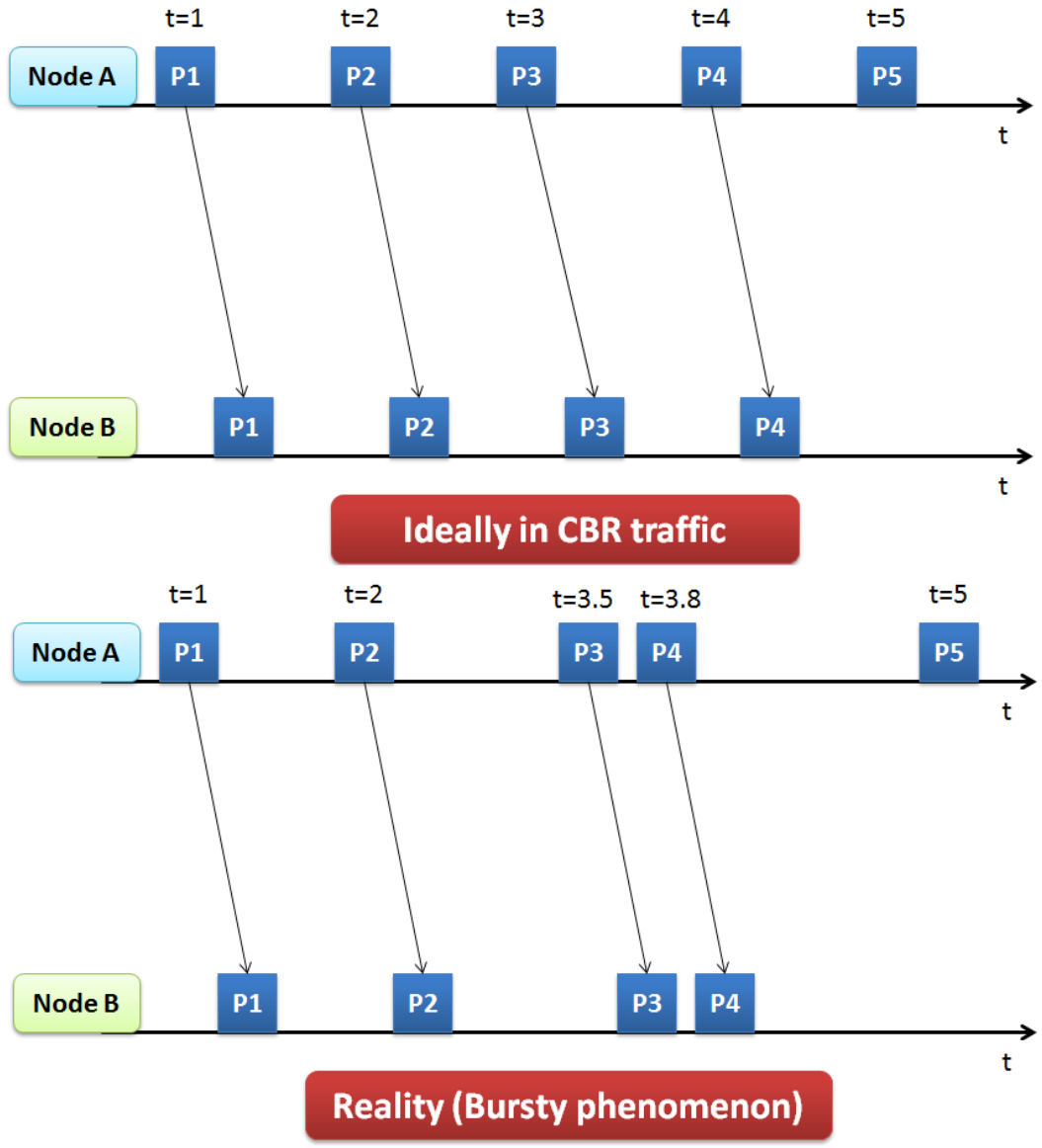

Figure 2.2: InterDeparture Time Accuracy

step was to find a statistical measurement that will give us quantitative information, the number of the packets that were transmitted not at the time that they supposed to. Hence, we created vectors for each TG on Matlab where we assigned values that were resulting from the abstract of the real IDT the preconfigured IDT. Finally, we plotted these vectors in "Box & Whisker" way. In the following chapters, a statistical analysis shows the results and assess the accuracy for all the tools under study.

2.2.3 CPU Consumption

Besides the throughput and interdeparture process performance we analyzed the resource utilization of each TG. In particular, we monitored the CPU consumption for each TG under different conditions. To do so, we used the tool *ps* [23] provided by the Linux distribution aimed at monitoring the status of each OS process (e.g. $ps - u - p$ "$processID$"). To keep track of the HW resources, we wrote a lightweight script that was running in parallel with the experiment logging the process status every half a second. Note that, even making an effort in having the lowest possible impact with the experiment itself, the values obtained included the HW usage of this script as well. However, all the TGs were evaluated under the exact same conditions and thus, the relative comparison among them is fair. We also crosschecked the values obtained by *ps* [23] with the ones obtained by using *htop* [18] (also present in most of the linux distributions) having practically equal results.

2.2.4 Sensitivity in Limited Resources

We run a set of experiments trying to emulate determined resource limitations (CPU, RAM and Linux Buffer) in order to analyze the sensitivity of TG in resource shortage. This HW limitations are a very common case, and the evaluation should be tested in a controlled environment like the one we are presenting here. This way, we can offer a qualitative study of the tools operating in HW-limited environments, e.g. Soekris, alix 2d2, aruba AP or linksys wrt54g wireless routers. We used the tool "stress" [2] in order to stress either the RAM memory or the CPU of the client/terminal:

$stress - c\,8$	for stressing the CPU
$stress - -vm\,4\ - -vm - bytes\,2048M$	for stressing the RAM memory

Hence, with the first command we stressed the CPU capacity, this is the case with forks of 8 processes, each of which spins in a tight loop calculating the sqrt() of a random number acquired with rand(), thus, the system achieves a load average up to an arbitrary value and finally, this holds until killed [14]. With the second command we stressed the RAM memory by allocating four memory processes, memory load (vmstat), and for each worker/process was loaded with 2048MB. Finally, in order to run experiments with reduced transmission Linux Buffer, we simply edited the files regarding to the send buffer size (i.e. wmem_max and wmem_default), they are located on the */proc/sys/net/core/* directory, for both files we reduced the buffer size to 1000 Bytes, thus, it was possible to transmit either one packet or couple of them depending of the packet size.

In the following Chapter, we present all the results related to the statistical measurements following the methodology described above: performance in throughput, inter-departure process and in resource usage, while in Chapter IV we show the results under limited HW resources conditions.

Chapter 3

Performance Assessment

This chapter shows the performance evaluation in ideal HW conditions carried out for different traffic configurations and for all the TGs under study.

3.1 Throughput Performance

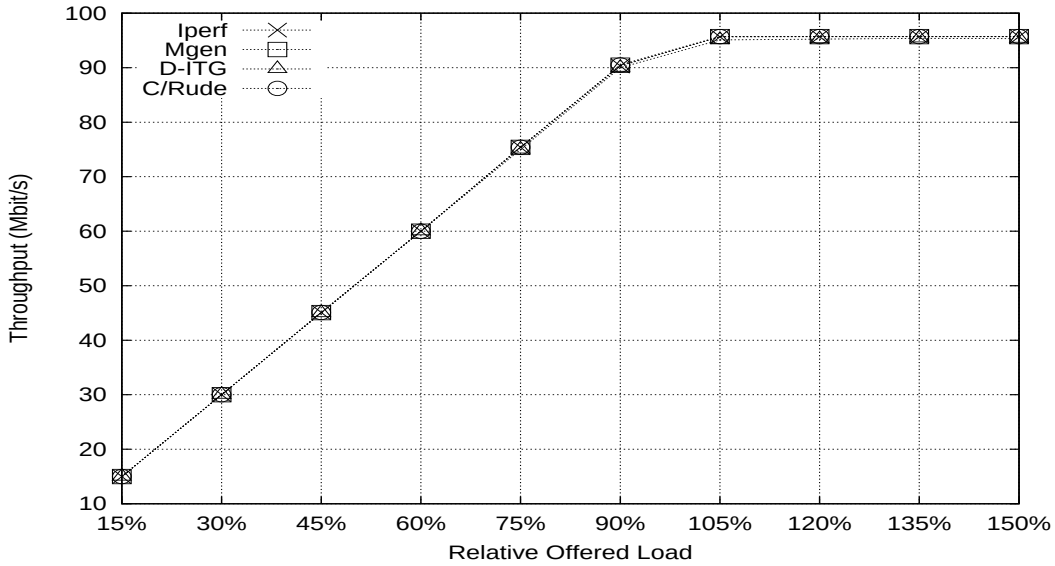

Figure 3.1: Throughput performance with payload of 1470Bytes

Figure 3.1 shows the throughput performance of TGs depicted for various offered loads and with a payload size of 1470 Bytes. Figure 3.2, in turn, shows the throughput performance under saturation conditions (i.e. transmision buffers are always backlogged), with a payload size of 147 Bytes, one tenth of previous set of experiments. In the first graph, the results show that all TGs achieve precisely the same throughput following always the expected theoretical values, an expected result taking into account the ideal HW conditions. However the second plot shows some

different results. On the one hand Iperf and C/RUDE achieve again the theoretical and expected values, which is the horizontal line at 69.014Mbps, but on the other hand, D-ITG and MGEN are slightly below the expected results. This indicates that the two latter find more difficulties when dealing with small packets at high packet rates.

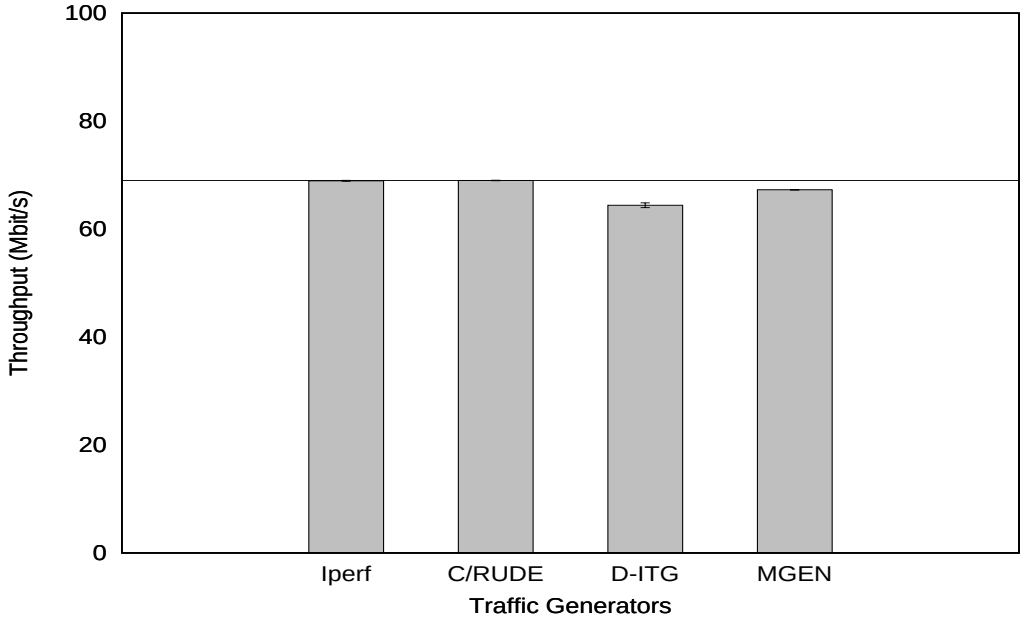

Figure 3.2: Throughput performance with payload of 147Bytes

3.2 Inter-Departure Time Performance

In this subsection, we present results regarding to the Interdeparture process accuracy of the transmitted packets. As we described in previous chapter, the goal of this procedure is to investigate the operational accuracy in terms of IDT of transmission. Figure 3.3 depicts some illustrative cases, packets sent every 196 and 130 microseconds, respectively, which correspond to 60% and 90% of the total expected saturation throughput.

In the figure 3.3, the results of the empirical CDF are shown. For all TGs most of the packets where sent with expected interdeparture times (as shown by a vertical line centered at the expected IDT due to the traffic profile of CBR). However, we can notice some cases of bursty behavior. By analyzing the case of 90% (i.e. a packet every 130 microseconds), we observe that C/RUDE operates more accurately compared to the rest TGs, an expected result, as C/RUDE claims to use better management of the timers [12]. This feature has a penalty in CPU consumption as we will analyze later on. Iperf has the worst behavior since the lines are not stable and it has big tails. Hence, from figure 3.3 we got a flavor of the IDT's variances of the TGs or in other words a qualitative view.

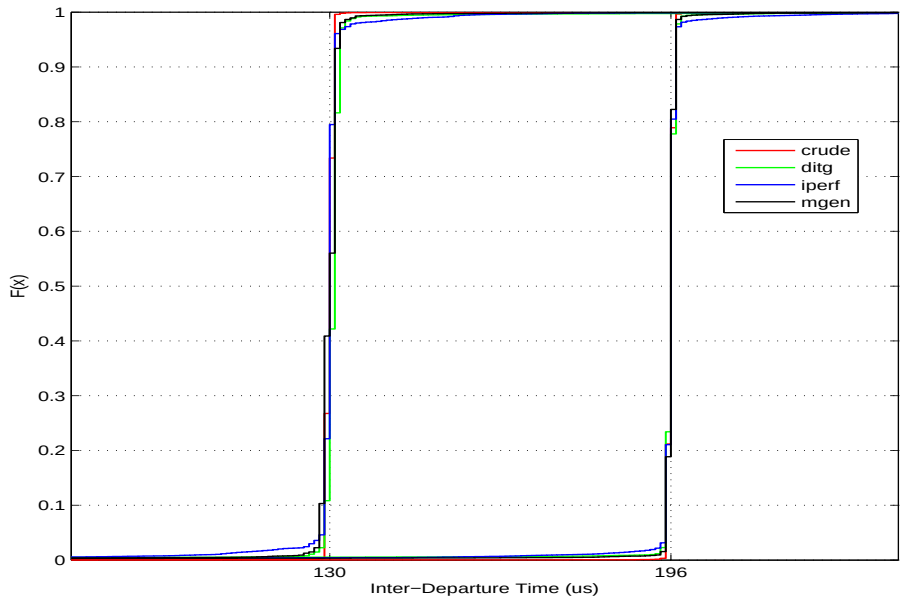

Figure 3.3: Empirical CDF for 60% and 90% of the Saturation

The figure 3.5 provides us with a quantitative view about the IDT and delay respectively. In box and whisker plot on each box, the central mark is the median, the edges of the box are the first quartile and third quartile (i.e. 25th and 75th percentiles respectively), the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually (i.e. the red points) [24]. The median for all TGs are at zero as it was expected; however IDTs of C/RUDE and D-ITG are spreaded around zero with very small value of variance while the other two TGs have a lot of outliers spreaded too far from the median. Hence, with this two figure we obtain a better overview about the delay performance of TGs.

3.3 CPU Consumption Performance

In this subsection, the results regarding to the CPU consumption are presented in Figure 3.4. There are two significant points to mention. The first one is about the very high CPU resources used by C/RUDE which is more than 90% of total CPU capacity. On the other hand, the rest of TGs consumes much less CPU resources. Iperf and D-ITG consume less than 10% and MGEN less than 20%. The second point that arises via these results is that the CPU consumption is getting higher with the increase of packet rates (as expected). When saturation conditions are overtaken, each generator behaves differently. For Iperf and D-ITG the CPU consumption decreases, while for the case of C/RUDE and MGEN we experiment a sudden decrease of CPU consumption but showing an increasing trend again when increasing the offered load.

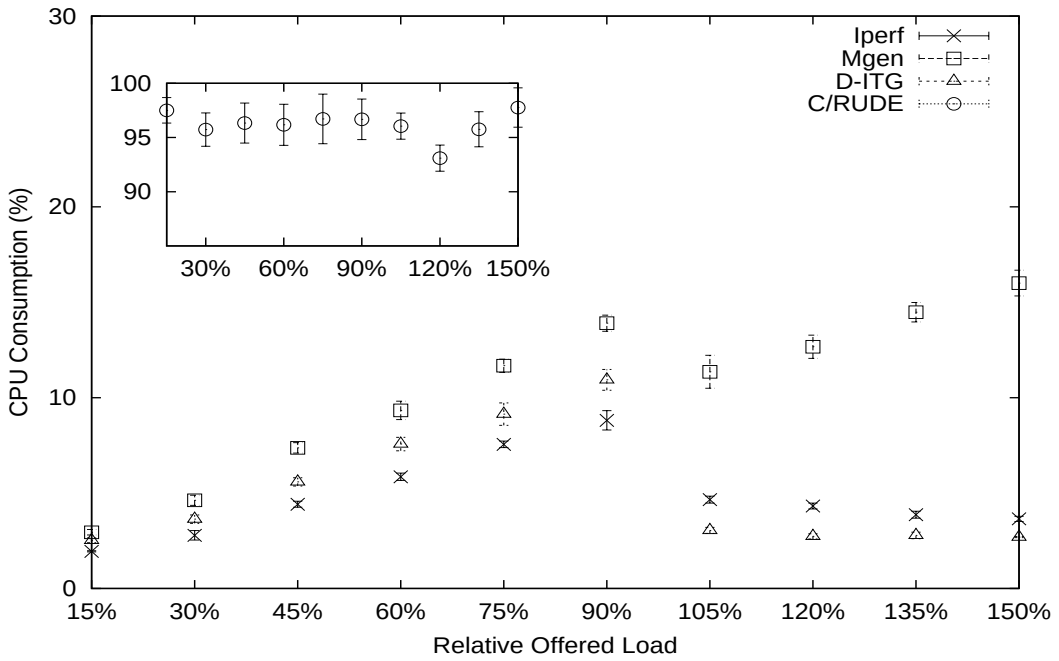

Figure 3.4: CPU Consumption with payload of 1470Bytes

It is worth mentioning again that the design of the C/RUDE tool was motivated by the goal of providing a generator with high accuracy in terms of timers management. To do so, instead of using limited system timers (as the other generators use), this tool first gets highest priority from the kernel scheduler, and uses timers with a resolution of one nanosecond. This greedy utilization of the resources are translated into a better accuracy, as shown in the previous subsection, but might be useless when dealing with limited HW resources.

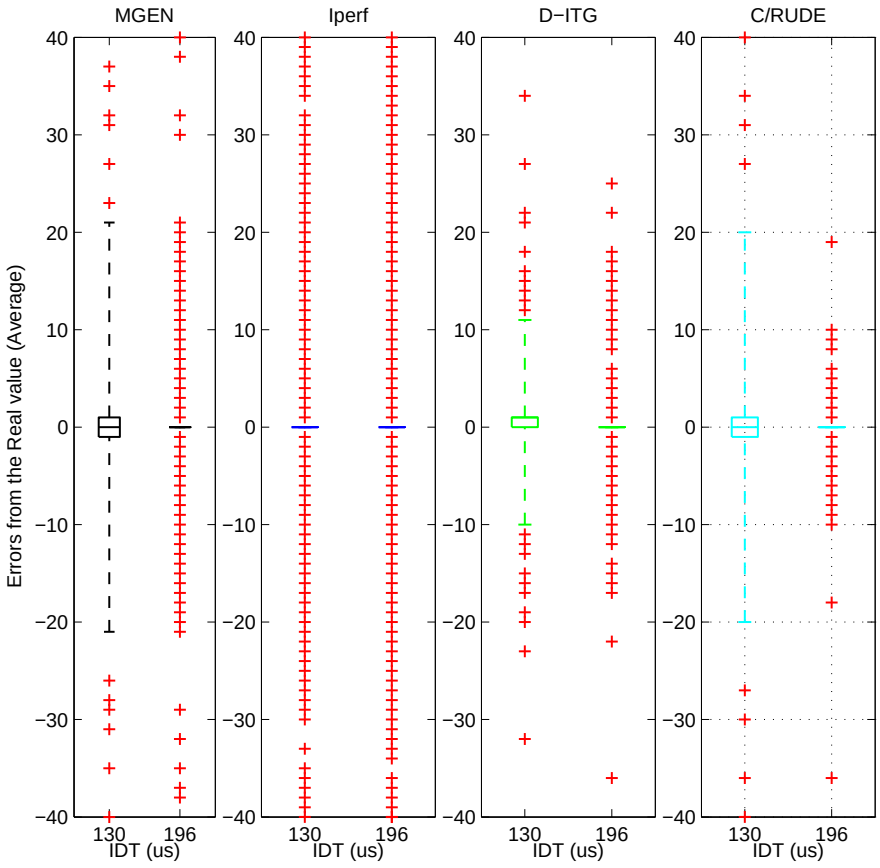

Figure 3.5: Box and Whisker for 60% and 90% of the Saturation

Chapter 4

Sensitivity Analysis of Resource Shortage

In this Chapter we present the performance analysis under different (and constrained) circumstances. First we will stress the CPU resources and test the behavior of the traffic generators, secondly we stress the memory resources (i.e. RAM), and finally we will reduce the Linux buffers.

4.1 Stressed CPU

The first result shows the throughput performance under limited CPU conditions. In this framework, the throughput obtained when using fairly large packet sizes (e.g. 1470 Bytes), and thus, relative small packet rate generation, were unaffected. Thus, we focus on the case of small packet sizes (147 Bytes) and high packet rates (i.e. 89285 packet/sec). Figure 4.1 shows the throughput obtained and compared with the obtained under ideal conditions (see previous chapter). We can observe that there exist an important impact to all TGs, especially to MGEN whose performance has been reduced by more than 25Mbps while for D-ITG and C/RUDE it is reduced by about 10Mbps.

Similarly as we showed in the previous chapter, we plot the resulting CDF of the Interdeparture process evaluation (see Figure 4.2). Operating under these conditions severely affects the performance of C/RUDE. The explanation now seems obvious: RUDE consumes a big share of HW resources to provide the claimed accuracy. If these resources are not available, it certainly (and severely) affects the performance. The rest of TGs do have relatively bad behavior, especially Iperf that presents more variability. It is worth pointing out that, even though all the tools are penalized by the low availability of HW resources (especially C/RUDE), their throughput performance remain as expected (for the case of big packet sizes).

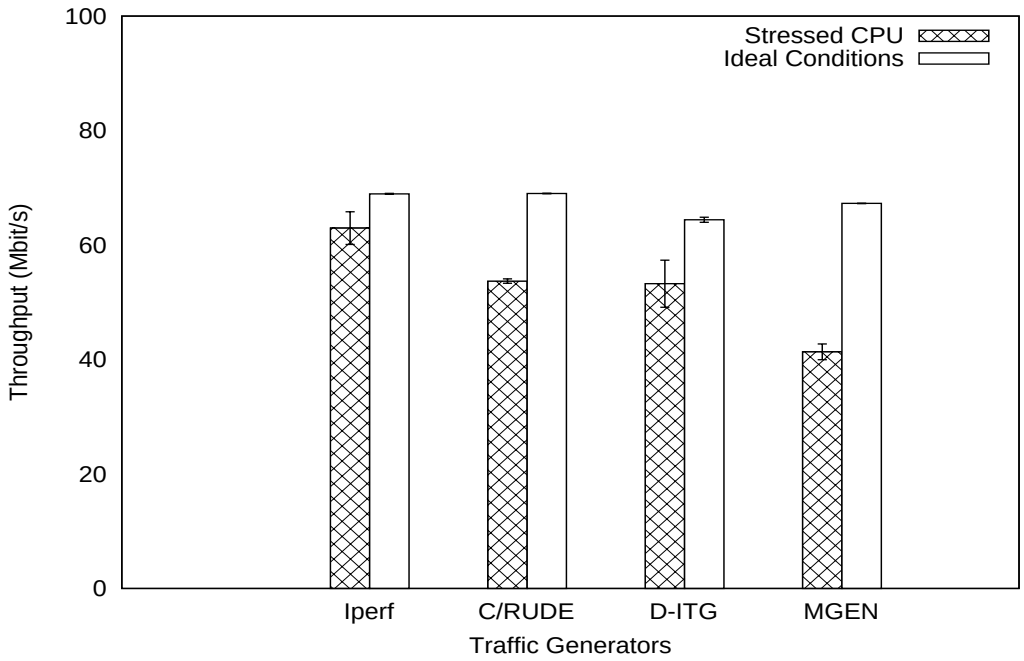

Figure 4.1: Throughput performance with payload of 147Bytes in Stressed CPU

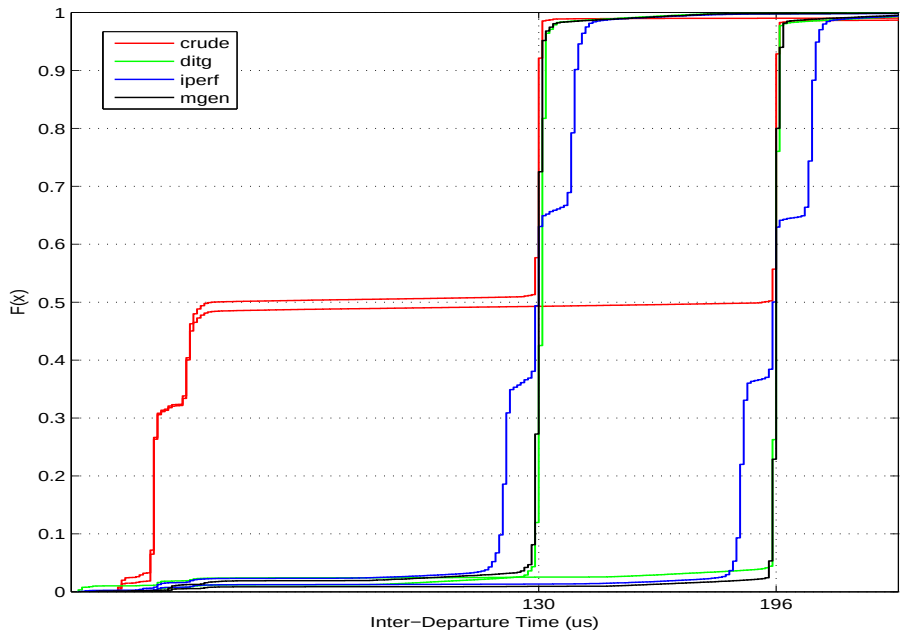

Figure 4.2: Empirical CDF for 60% and 90% of the Saturation in Stressed CPU

4.2 Stressed RAM

Prior to the evaluation with stressed memory resources, we show in Table 4.1 the memory size of the each TG's binary file.

Table 4.1: The Binary Size of TGs

Iperf	ITGSend	MGEN	RUDE
63.3 kB	271.4 kB	843.5 kB	21.9 kB

In this scenario, D-ITG seems to suffer severely since it achieves less than 20Mbps of throughput while C/RUDE and MGEN are following precisely the expected values in case with payload of 1470Bytes, Iperf, in turn, achieves less than the expected as well (see Figure 4.3). Similar phenomenon can be observed in the case where the payload size is 147 Bytes (see Figure 4.3, the throughput performance it is compared with the results of the ideal conditions). C/RUDE and MGEN follow their previous performance, Iperf achieves slightly bellow comparing with value of the original scenario and finally, D-ITG has the worst behavior since it reaches less than 10 Mbps.

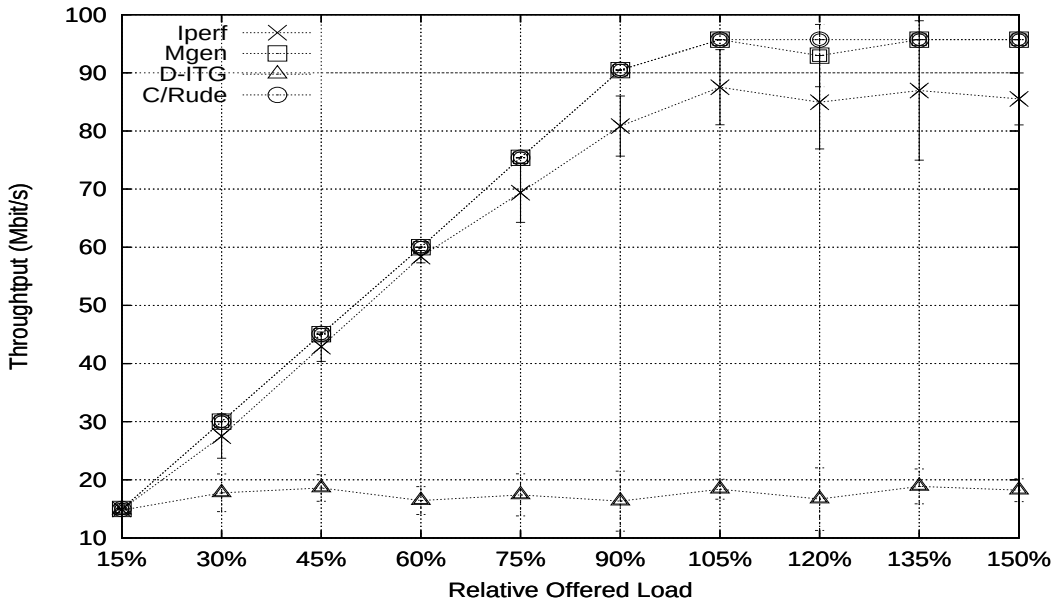

Figure 4.3: Throughput performance with payload of 1470Bytes in Stressed RAM

In the figure 4.5 the performance of the TGs in terms of IDT is depicted. The TGs' behavior is respectively to their throughput performance. In particular, D-ITG has almost non vertical line in the graph, thus, there is very high percentage of burdty. Furthermore, Iperf shows bad performance as well, while C/RUDE and MGEN in turn, perform better than the other two but still they have long tails, especially MGEN.

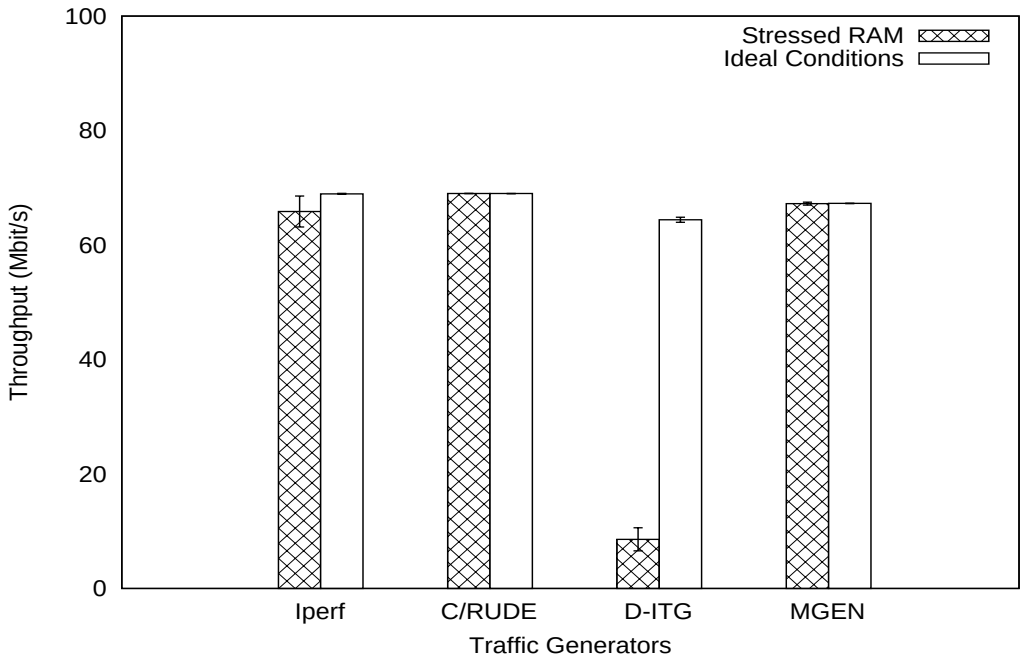

Figure 4.4: Throughput performance with payload of 147Bytes in Stressed RAM

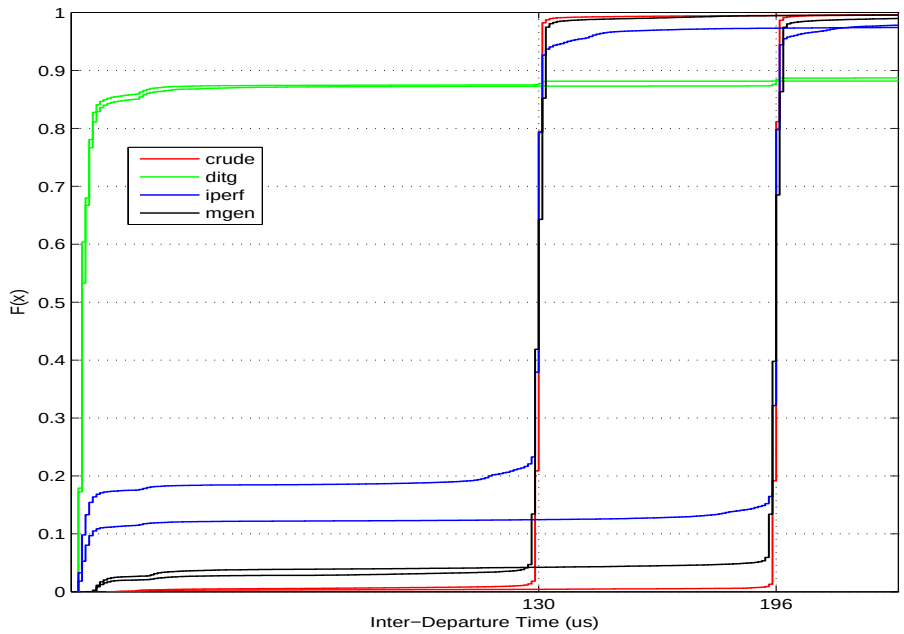

Figure 4.5: Empirical CDF for 60% and 90% of the Saturation in Stressed RAM

4.3 Limited Buffer

The last scenarios of this chapter is with limited transmission buffer. In this scenario where we reduced the transmission buffer of the kernel, in chapter II we gave a detailed description of how we did it. Most of TGs apart from D-ITG achieve precisely the throughput performance of the ideal conditions. D-ITG after certain point, of 75% of the saturation point, stops following the theoretical values and starts decreasing trend. In other words after a certain packet rate cannot handle the transmission, thus, this behavior is well depicted in the figure 4.6.

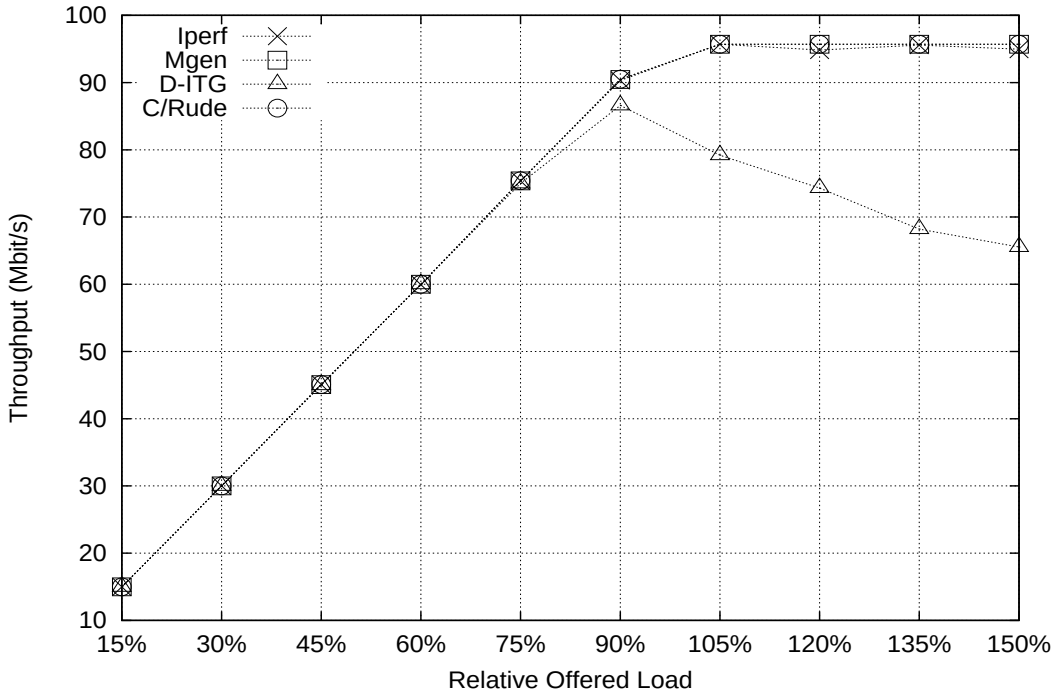

Figure 4.6: Throughput performance with payload of 1470Bytes in Limited Buffer

On the other hand the results related to the IDT (see Figure 4.7), the behavior of D-ITG is getting very bad with the increase of the packet rate (or when the IDT decrease). The rest TGs achieve precisely the expected values.

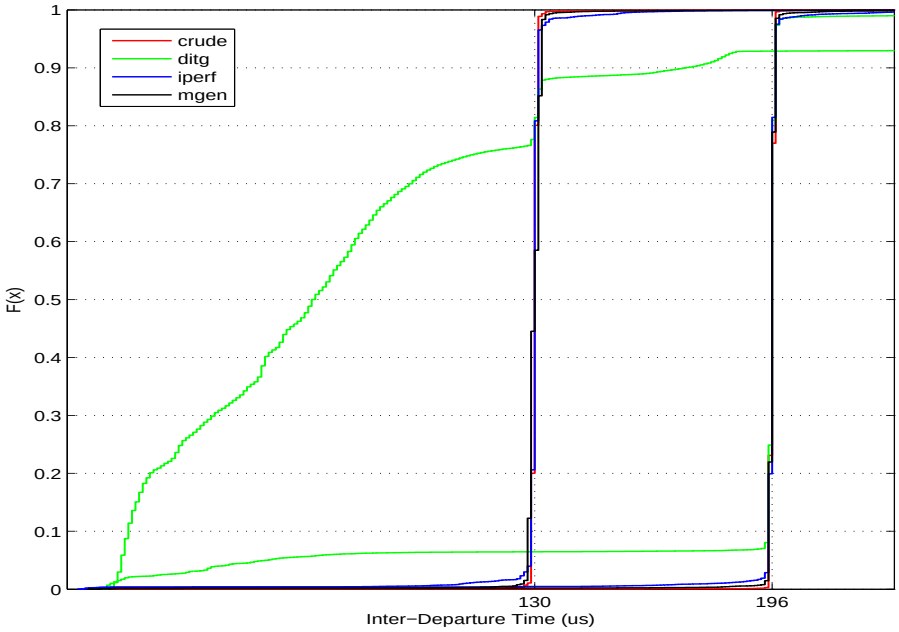

Figure 4.7: Empirical CDF for 60% and 90% of the Saturation in Limited Buffer

Chapter 5

Conclusions and future work

In this thesis we evaluated the performance of four representative (and most popular) network traffic generators in a laboratory environment under a wide set of experiments. The main contribution of this study was to categorize the TGs according to their features, and to investigate their performance (i.e. throughput, CPU consumption and Inter-Departure time accuracy) in wired connection (i.e. Fast Ethernet of 100Mbps) using different metrics. Hence, the goal of this work was to provide researchers and IT engineers an answer regarding the TG tool they should use according to the experiment they need to run.

After of thorough research on the state of the art and after of running big number of experiments we came with the following conclusion, *there is not a real winner among the TGs*, since each TG has it own advantages and disadvantages.

Characteristics C/RUDE is the one that provides a more complete report of metrics (i.e. Throughput, Delay, Jitter, Packet Loss and Interarrival statistics). Excluding C/RUDE, all TGs support both TCP and UDP protocols, they are capable of running on Windows platforms (not only Linux) and support IPv6. Furthermore, D-ITG, differently from the rest of TGs, provides support for a wide set of other protocols such as SCTP or DCCP in the transport layer, ICMPv4 and ICMPv6 in network layer and also protocols for application layer like Telnet, DNS and VoIP. Moreover, MGEN is capable to transmit packets in multicast and broadcast mode in addition to unicast while among the rest, only Iperf supports multicast; however it is enough complicated in use and does not provides the results directly. Finally, Iperf is easy to use but it has a signifficant drawback since it only supports CBR while C/RUDE supports an additional traffic profile called TRACE. MGEN supports CBR and four more: Poisson, Burst, Jitter and Clone. D-ITG, in turn, supports more traffic patterns than any other (i.e. Uniformly, Exponentially, Pareto, Cauchy, Normal, Poisson, Gamma, Weibull and Burst).

Throughput The results for the case of various offered loads and with a payload size of 1470 Bytes, show that all TGs achieve precisely the same throughput following always the expected theoretical values, an expected result taking into account the ideal HW conditions. Hoewever, in the second case with a payload size of 147 Bytes, one tenth of previous set of experiments and under saturation conditions (i.e. transmision buffers are always backlogged), the results show different behavior

among the TGs. On the one hand Iperf and C/RUDE achieve again the theoretical and expected values, but on the other hand, D-ITG and MGEN are slightly below the expected results. Furthermore, depending the scenario (Stressed CPU, RAM or Limited buffer) our results show different performances for the TGs.

Inder-Departure Time C/RUDE operates more accurately compared to the rest TGs, an expected result, especially in cases where the packet rates are close to the saturation point. Furthermore, by analyzing box and whisker figure, the median for all TGs are at zero as it was expected; however IDTs of C/RUDE and D-ITG are spreaded very close to zero while the other two have a lot of outliers spreaded too far from the median.

CPU Consumption C/RUDE is the one that consumes the most CPU capacity (i.e. over 90%) compared with the rest, followed by MGEN less than 20%. Iperf and D-ITG consume less than 10%. As we mentioned in chapter III C/RUDE was motivated by the goal of providing a generator with high accuracy in terms of timers management, thus, this tool first gets highest priority from the kernel scheduler, and uses timers with a resolution of one nanosecond. Hence, this greedy utilization of the resource is translated into a better accuracy.

As for future works, would be nice to run the same experiments with TCP as a transport protocol in order to obtain a better overview. Moreover, experiments on non powerful machines in terms of CPU and RAM memory (e.g. Soekris) would be very important to check the whether indeed some TGs will not operate properly since they need high resources. Furthermore, it is interesting to see results of experiments that run in fast ethernet of 1Gbps, our guess is that there would be a difference among the TGs even in the case of ideal conditions.

Finally, our suggestion to researchers and IT engineers about which TG to choose is the following two hints. First step, define the type of experiment (e.g. Throughput Delay measurement.), and as a second step check the hardware characteristics, how powerful are the machines that the experiments will take place. The Table 5.1 contains a qualitative summary of our research.

Table 5.1: The Qualitative Summary of TGs

TG	Throughput	Delay	Resource Consumption	Impact of Resource Shortage
Iperf	Very Good	Poor	Good	Bad
C/RUDE	Very Good	Very Good	Very Bad	Good
D-ITG	Good	Good	Good	Bad
MGEN	Good	Good	Bad	Good

References

[1] Perl. http://www.perl.org/, July 2012.

[2] Amos Waterland. stress. http://weather.ou.edu/ apw/projects/stress/, July 2012.

[3] P. E. A. N. R. G. at Naval Research Laboratory. MGEN, The Multi-Generator Toolset:. http://cs.itd.nrl.navy.mil/work/mgen/, Mar 2012.

[4] P. E. A. N. R. G. at Naval Research Laboratory. TRPR, Tcpdump Rate Plot Real Time. http://cs.itd.nrl.navy.mil/work/proteantools/, Mar 2012.

[5] D. A. S. T. D. at the National Laboratory for Applied Network Research (NLANR). Iperf. http://iperf.sourceforge.net/, Mar 2012.

[6] S. Avallone. Mtools. http://www.grid.unina.it/grid/mtools/, Mar 2012.

[7] S. Avallone, D. Emma, A. Pescapeè, and G. Ventre. A Distributed Multiplatform Architecture for Traffic Generations. To appear in Proc. of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS04), July 2004.

[8] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre. D-ITG distributed internet traffic generator. First International Conference on the Quantitative Evaluation of Systems, 2004.

[9] A. Botta, A. Dainotti, and A. Pescapè. Multi-Protocol and Multi-Platform Traffic Generation and Measurement. INFOCOM 2007 Demo Session, May 2007.

[10] Candela Technologies. LANforge. http://www.candelatech.com/, Apr 2009.

[11] D-ITG. Other Traffic Generators. http://www.grid.unina.it/software/ITG/link.php.

[12] F. . (former Faster Pro) at Tampere University of Technology. RUDE & CRUDE, Real-time UDP Data Emitter and Collector for RUDE. http://rude.sourceforge.net/, Mar 2012.

[13] E. Gamess and K. Velasquez. IPv4 and IPv6 Forwarding Performance Evaluation on Different Operating Systems. XXXIV Conferencia Latinoamericana de Informatica (CLEI 2008), Sep. 2008.

[14] G. General Public License. Stress 1.0.1. https://laas.mine.nu/uclibc/stress-1.0.1/doc/stress.html, July 2012.

[15] G. General Public License (GPL). Network Traffic Generator. http://sourceforge.net/projects/traffic/, Jan 2003.

[16] Google. Google Scholar. http://scholar.google.com/, July 2012.

[17] Henning Schulzrinne. Traffic Generator. http://www.cs.columbia.edu/ hgs/internet/traffic-generator.html.

[18] Hisham Muhammad. htop. http://htop.sourceforge.net/, July 2012.

[19] IEEE. IEEE Xplore. http://ieeexplore.ieee.org/Xplore/guesthome.jsp, July 2012.

[20] S. International and U. P. C. for Experimental. TG2002. http://www.postel.org/services.html, Jan 2002.

[21] R. Jones. Netperf. http://www.netperf.org/netperf/, Mar 2012.

[22] S. S. Kolahi, S. Narayan, D. D. T. Nguyen, and Y. Sunarto. Performance Monitoring of Various Network Traffic Generators. XXXIV Conferencia Latinoamericana de Informatica (CLEI 2008), Sep. 2008.

[23] Linux Unix Command. ps. http://linux.about.com/od/commands/l/blcmdl1_ps.htm, July 2012.

[24] MathWorks. Boxplot. http://www.mathworks.es/help/toolbox/stats/boxplot.html, July 2012.

[25] Nuts About Nets. NetStress. http://nutsaboutnets.com/netstress/, Sep 2008.

[26] T. U. of Michigan. UDP Generator. http://www.citi.umich.edu/projects/qbone/generator.html, Mar 2012.

[27] Protocol Testing. Network Packet/Traffic Generator Tool. http://www.protocoltesting.com/trgen.html.

[28] Sally Floyd. Traffic Generator for Internet Traffic. http://www.icir.org/models/trafficgenerators.html.

[29] Z. Telecom. IP Traffic. http://www.zti-telecom.com, Mar 2012.

[30] G. N. University of Agder. WLAN Traffic Visualizer. http://sourceforge.net/projects/wlantv/, Dec 2008.

[31] K. Velasquez and E. Gamess. A Comparative Analysis of Network Benchmarking Tools. I Proceedings of the World Congress on Engineering and Computer Science (WCECS) 2009, Oct 2009.

[32] Wireshark. Tshark. http://www.wireshark.org/docs/man-pages/tshark.html, July 2012.