How to generate realistic network traffic ?

Antoine Varet, Nicolas Larrieu

How to generate realistic network traffic?

Antoine VARET and Nicolas LARRIEU

ENAC (French Civil Aviation University) - Telecom/Resco Laboratory

ABSTRACT

Network engineers and designers need additional tools to generate network traffic in order to test and evaluate, for instance, application performances or network provisioning. In such a context, traffic characteristics are the most important part of the work. Indeed, it is quite easy to generate traffic, but it is more difficult to produce traffic which can exhibit real characteristics such as the ones you can observe through the Internet. With the lack of adequate tools to generate data flows with "realistic behaviors" at the network or transport level, we needed to develop our tool entitled "SourcesOnOff". The emphasis of this article is on presenting this tool, explaining how we implemented it and outlining the methodology it follows to produce traffic with realistic characteristics.

Keywords *Traffic generation, Network experimentation, Network performance analysis, Network measurement*

1. INTRODUCTION

Network engineers and designers need additional tools to generate network traffic in order to test and evaluate application performances or network provisioning for instance. In such a context, traffic characteristics are the very important part of the work. Indeed, it is quite easy to generate traffic but it is more difficult to produce traffic which can exhibit real characteristics such as the ones you can observe in the Internet. With the lack of adequate tools to generate data flows with "realistic behaviors" at the network or transport level, we needed to develop our tool entitled "SourcesOnOff[1]". In this paper, "realistic behavior" means to generate traffic similar to the traffic a network administrator can capture on his network backbone, end-systems, etc...

Previous research conducted us to design new methodologies and tools to generate network data traffic as close as possible to what we can find on Local Area Networks (LAN) and on the Internet. It can be necessary for instance to evaluate performances of new network entities. In this specific context, it is mandatory to face them to generated traffic with characteristics as close as possible as the Internet traffic. However, depending on what you mean by "Internet network" and where you perform the study, measurements may completely differ: Internet cannot be solely characterized with a small set of parameters such as some mathematical distributions and additional factors. Indeed, there is not currently one unique mathematical modeling able to embrace the different characteristics and the complexity of the Internet traffic [1]. Anyway, Internet traffics hopefully show some common properties and trends that help to its modeling. For instance, an Internet Service Provider (ISP) providing access for software engineering companies manage a different profile of data communications than an ISP for private individuals [2]. However, there are some common characteristics between both profiles. The goal of the tool we have developed is to handle most of Internet traffic profiles and to generate traffic flows by following these different Internet traffic properties.

The rest of this paper is structured as the following. The first section presents the mathematical model this tool is based on for traffic generation: the ON/OFF sources. We chose to consider different stochastic processes in order to model the complexity of the original traffic we want to replay. General approaches consider only one law for the On process and another law for the Off process. In our approach, we are able to consider several laws and to combine their effects to model accurately the original behavior we analyzed in the real data. We then select the right parameters to consider as inputs for our SourcesOnOff tool. This approach gives really good traffic characteristics and, consequently, the generated traffic is really close to reality as results presented at this end of this paper demonstrate it. In the second section, we present why the other tools currently available were not usable in our case. The third section describes how our tool works plus some specific points the user should know about its implementation. The fourth section is dedicated to the validation of our tool with different methods in order to conclude if the generated traffic has the same properties than the original one. We have, in particular, investigated the traffic characteristics generated by our software and we demonstrate that it is able to generate network traffic with realistic properties such as these we can capture on the Internet. Different parameters have been considered to match generated traffic characteristics with the original traffic. First, we considered classical traffic parameters such as throughput, delay or losses but we have also investigated more advanced statistical parameters such as the correlation level of generated packets or their long range dependence (by computing the Hurst factor).

1.1 How to characterize an "Internet-like" profile?

Different studies were published with results related to Internet-like traffic characterization: the word "Internet" can cover very different profiles, but some common trends can be highlighted. Thus, we will examine two of them: high variability and self-similarity, respectively called the Noah and Joseph Effects.

High variability is characterized by an infinite mathematical variance and means that sudden discontinuous changes can always occur. Some mathematical distributions like Pareto and Weibull are heavy-tailed (i.e. the tail of the distribution is not exponentially bounded) and thus can be used to generate sets of values with high

[1] We would like to thank Benoit Sainct (MS student) for his work in implementing statistical tools for traffic characterization.

Contact email: Nicolas.Larrieu@enac.fr

variances and also high-variability. Different studies have shown that classical distributions such as the Gaussian and the Poisson distribution have failed to reproduce a LAN-like throughput [1, 3]. One of them [1] proposes to use Pareto's and Weibull's heavy-tailed distributions and proved that the generated traffic shows properties closer to real network traffic characteristics than other mathematical distributions.

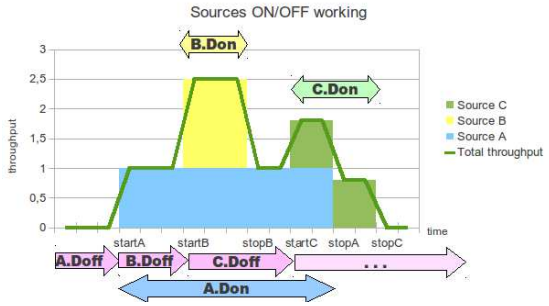

Figure 1: Emission process of ON/OFF traffic sources

Self-similarity is defined by a long-range dependence characteristic, which means there are bursts of traffic any time over a wide range of time scales. In other words, a small sub-range of values is "similar" to the whole range of values. W. Willinger found in [3] a relation between self-similarity and high variability for Ethernet Local Area Network (LAN) throughputs: in particular, he showed that the Noah and the Joseph effects are linked with a degree-1 polynomial relation. In other words, using ON/OFF sources with heavy-tailed distributions causes the traffic streams to be highly variable and, consequently, the aggregation of these streams to be also self-similar and highly variable.

1.2 ON/OFF sources to generate realistic throughputs

W. Willinger proposed in [3] to generate data following packet-train models, i.e. with strictly alternating, independent and identically distributed ON- and OFF-periods and proved that these **ON/OFF sources** generate data similar to experimental measurements on real networks. A source corresponds to one network flow. The flow is associated with a source and destination couple and also with data, transmitted as a set of packets called a train of packets. In our case, we use the transport protocols TCP and UDP. The data stream is thus exchanged as a set of IP packets. The source is associated to a departure time and a duration time. The departure time of any source is computed with the departure time of the preceding source plus a random duration. This randomness follows the user-defined "**Doff distribution**" (also named "*distribution of inter-train durations*"). The first source starts at the beginning of the process.

Duration times should be similar for all sources. They should follow a user-defined random distribution called "**Don distribution**" (also named "*distribution of train duration*" or "*distribution of flow duration*"). This Don distribution is independent of the Doff distribution. For feasibility constraints and to reduce the complexity of implementation, we use the Don distribution to generate random values, not for time duration in seconds but for **quantity of transmission in bytes**. Because of TCP congestion control mechanisms, source durations are correlated with quantities of transmitted data.

We can summarize the ON-OFF source generation algorithm we use in our tool by:

For n from 1 to infinite do:

/ start a new source here */*

1. *Don_value* **:=** *get a random value of distribution Don;*
2. *Doff_value* **:=** *get a random value of distribution Doff;*
3. *Wait for the duration Doff_value*
4. *Start the transmission of Don_value bytes to a remote host (do not wait the end to perform the next iteration, loop as soon as data are sent into the transmission buffer)*

End_For

Algorithm 1: Traffic generation with ON/OFF sources

In algorithm 1, the Don_value is the length (in bytes or multiples of bytes) of each data flow, also called a "train". This train is exchanged between the local (transmitter) system and a remote (receiver) system. The Doff_value (in seconds or more often in milliseconds) is a duration called the inter-train distance (i.t.d). This i.t.d. represents the time between two consecutive flow creations. The Doff_values and the Don_values are completely independent. They are issued from random number generators, which follow respectively the "Don" and the "Doff" distributions. These distributions should be heavy-tailed to ensure the self-similarity of the generated data throughput [3]. Distribution parameters are defined by the user on the command-line. Figure 1 illustrates an example of the total throughput we can expect on the network following the ON/OFF sources generation process.

2. STATE OF THE ART OF TRAFFIC GENERATORS

2.1 Network simulators

Different tools are available to simulate networks and their behaviors, with integrated ON/OFF sources (for instance, NS-2 [4]). Most of them provide "Internet-like" flow generators, some of them based on ON/OFF source generation processes. However, the aim of our research work was to face new network systems to real traffic. It means we wanted to perform our system evaluation in real time and do not want to evaluate a model of our system. Simulators cannot do this kind of real time experiments, only emulators can do it. However, it is complex to deal with most network emulators as they are complex to install, to configure with the real network environment, and consume as well many CPU resources. This is why we searched for and studied tools not to simulate or emulate but to generate a realistic traffic load on real networks.

2.2 Traffic replay tools

Some software enables the user to replay previously captured network traffics. This is done in two steps: the user captures data on the networks he wants to reproduce (with tools called "sniffers" such as the reference tool Wireshark [5]), then the user needs to replay the captured traces, i.e. by retransmitting the sniffed packets in the same order, and separated with the same delays as these measured during the capture (cf. the "tcpreplay"[2] tool for instance). Harpoon[3] is another existing Open Source flow-level traffic generator, but this tool requires the user to define flow per flow the data weight he wants to transmit on the network. The automatic generation of the network profile is limited to constant and uniform distributions. Moreover, Harpoon seems to be not maintained since 2005. This is why we did not consider this tool for our experiments.

This traffic replay process has different advantages. The two steps may be done independently by two different users and at any time (as long as the first step is started before the second one). The capture may be filtered before being replayed and replay parameters may be set up. However, replaying has also important drawbacks. Indeed, before replaying any network trace, the user must make or acquire the capture he wants to replay. Most of the time, this is a difficult task: user privacy issues restrain the administrators in allowing captures on the router they administer; often ISPs do not want to extract advanced statistics from their networks… The public data we have found on Internet are often so anonymized that they not longer contain the useful information we need to replay them. Moreover, if you continuously replay the same trace you will reproduce periodically the same "events" (throughput bursts, specific packet sequences or behaviors…).

2.3 Network throughput estimation tools

This is why different techniques and tools have been elaborated to load a network and evaluate its capacity. One of the existing tools we have studied is "iperf" [6]. It can generate TCP or UDP flows to load the network. In TCP mode, an iperf client transmits to the server an infinite quantity of data through one TCP flow. After a user-defined duration, the iperf tool aborts the TCP connection and prints on the screen different statistics and the total quantity of data it succeeded in transmitting correctly.

A network capture shows often that mostly all the network resources are used by the TCP flows [2]. Indeed, the TCP protocol efficiently exploits the network and transmits data in an optimal time span. However, in LAN captures, external and unexpected events interact with the flow, they generate segment delays and losses impacting the TCP connection which becomes longer than the optimal time span expected. This drawback affects most of methodologies based on studies using only one TCP flow. We can cite, for example, when the tester uses an SSH or an FTP connection to transmit a big file and consequently to artificially load the network.

When configured to transfer multiple TCP streams, iperf starts the different flows simultaneously at the beginning of the program. This behavior is not realistic because in Internet later flows are penalized at their start by the flows already established. Moreover, this tool "iperf" performs measurements on "long" TCP connections (100+ segments), allowing the TCP congestion control mechanism to adjust the data throughput efficiently. However, in most ISP studies the TCP connections are very numerous and the majority is short (e.g. in study [6] 81% of TCP connections carry less than 310 bytes), disabling the congestion control mechanisms optimizations. Consequently, congestion control mechanisms do not have the time to optimize the connection. In UDP mode, the client transmits data periodically: the period is computed depending on the user-defined quantity of data per second to transmit. However, for the same reason than previously, a single flow with constant throughput is not realistic [1, 2].

Consequently, we have searched for other tools to generate data throughput: BWPing [7], NetPerf [8]… All differ more or less on the set of supported protocols, on the maximum admissible throughputs and on the statistics printed on the screen for the user. However, all have the same drawbacks as explained above. This is why, in the rest of this article, we chose the tool "iperf" as a comparison reference for our SourcesOnOff tool, because most measurement tools are based on the same assumptions and work in a similar manner: one or multiple identical and simultaneous TCP and UDP flows in order to load the network at its maximum available capacity.

3. THE SOURCESONOFF TOOL

We did not find in the state of the art we performed any tool to generate data flows based on exact ON/OFF sources. This is why we developed this one in C language and validated it with the Debian Operating System. The tool we propose is free and Open Source, under the General Public License v3 (GPLv3). Source code can be downloaded at http://www.recherche.enac.fr/~avaret/sourcesonoff.

3.1 Generation of the random values

ON/OFF sources are based on the generation of random values following well-determined distributions. The following distributions are currently implemented in the program. The **Uniform distribution** is based on the drand48() and the random() Linux functions, with corrections to ensure an exact uniformity on any range. The **Normal/Gaussian distribution** is computed with the help of the Box-Muller transformation. We use the Knuth's algorithm for the **Poisson distribution**. The **Pareto, Weibull and Exponential distributions** use internally a transformation on the uniform distribution. The generation process of all these distributions have been statistically validated with the R statistical software [9] and its comparison function qqplot(). An additional pseudo-distribution is available: the **Constant** distribution. In this

[2] Tcpreplay website: http://tcpreplay.synfin.net/

[3] Harpoon website: https://github.com/jsommers/harpoon

case, all generated values are equal to a user-defined constant. This method enables the user to generate a more predicting behavior (note that this behavior is similar to iperf behavior in UDP mode).

Multiplying factors enables users to convert the randomly generated values into Bytes (for Don distributions), nanoseconds (for Doff distributions) and their multiples (kB, MB, GB, us, ms, s…). Distributions may be bounded by minimum and maximum user-defined values. Our tool enforces also user-defined minimum and maximum values for the generated random numbers, by increasing values lower than the minimum boundary and decreasing values greater than the maximum boundary.

3.2 Generation of the network flows

First of all, different sets of Don and Doff random values are generated. They are then used for data communications. These data flows can be fully parameterized in order to enable the user to refine the tool behavior. Different sets of sources may run simultaneously, each set is associated with independent Don and Doff distributions and parameters. Contrary to programs like iperf, this tool is only intended to generate data. It was not developed to provide itself advanced statistics on the network: we use our tool to generate background traffic and we use additional passive and active measurement tools to evaluate network performances and collect statistics.

3.3 Statistic profile extraction from a real traffic trace

Before experimenting traffic generation, we have to define what kind of traffic we want to generate. This step may be done arbitrarily by choosing appropriate values or by selecting values from existing studies like [1, 2]. We chose an alternative solution: given that we belong to a research entity, we have easily access to real data. Thus, we asked to our local network administrator to capture the entire incoming and outgoing traffic, generated by people from the university (students, administrative people, teachers and researchers) on our local area network. The data were captured on the firewall protecting the access link between our LAN and other networks: REMIP and RENATER which are our links to the Internet. Thus, we captured all data from our LAN to Internet and all associated responses.

We captured different network traces: between 10 minutes and 10 hours. We chose to capture traffic only on working days and between working hours given that the point of presence is at the output of a research entity and so, the main network activity is spread from 8AM to 8PM (Monday to Friday). For concise purposes, we will describe in the next subsections only one of the different traffic traces we collected, but other captures showed the same conclusions. Thus, in the next subsections we analyze a set of 9 millions of IPv4 packets (97.7% of TCP, 2.2% of UDP and 0.1% of ICMP), collected during 10 hours between 8:00AM and 6:00PM, Tuesday the 29th of January, 2013. UDP datagrams are negligible in our case (less than 2.2% of our traffic in packet numbers, less than 0.5% of our traffic in bytes), so we replayed only TCP sources and we will just model and reproduce the TCP traffic in our study case (cf. section 4).

We used the tool tcpdump [10] to capture the raw packets and then the tool ipsumdump [11] to ensure the anonymity. We finally conducted a complete statistical analysis with bash and R scripts. These tools enabled us to retrieve different statistical characteristics of the captured data. This process is detailed in the next subsection.

3.3.1 Statistic profile extraction process

Previous research work (such as [1, 2, 3 and 4] for instance) have demonstrated that one unique statistical law cannot figure out all the complexity of an original Internet traffic trace. This is why we chose to decompose and to model the Internet traffic trace we want to replay by using several different statistical laws. Thus, we need, firstly, a **decomposition algorithm** and, secondly, a **distance criterion** to evaluate the differences between real original data and data generated by our tool. This distance criterion will help us to select the best statistic laws to decompose original data.

3.3.1.1 Traffic trace decomposition

We developed an algorithm able to detect a lack of continuity in any data we want to characterize. This algorithm is mainly based on the quantmod[4] tool developed by Jeffrey A. Ryan. Based on this algorithm, we are able to select different basic distributions (Weibull, Pareto, Exponential, Gaussian…) that we can combine to reproduce the whole complexity of the original data we want first to characterize and finally to replay. To assess the distance between selected statistical distributions and real data, we used the Bayesian Information Criterion which is introduced in the next subsection. We are thus capable of combining the different values of the traffic to generate by considering those different distributions in the SourcesOnOff tool.

3.3.1.2 BIC (Bayesian Information Criterion) distance assessment

The goal of this step is to quantify the statistical distance that exists between our original data and the data generated by our tool. To compute this distance we introduce the Bayesian Information Criterion [12]. This criterion is computed as

BIC = k * ln(n) – 2 * ln(L), where:

- *n* is the size of analyzed data;
- *L* is the likelihood of the model (Weibull, Pareto, Exponential…) regarding the different original data;
- *k* is the total number of estimated parameters.

The final goal of this comparison is to select the smallest BIC (minimum BIC value is -∞) according to the different candidate statistic profile (Weibull, Pareto, Exponential, Gaussian…). Thus, the tool can conclude that the selected model (which might be a composition of different distributions) is the closest from the original data.

[4] Quantmod website: http://quantmod.r-forge.r-project.org

4. SOURCESONOFF VALIDATION:

The objective of this section is to validate the traffic profile generated by our tool by verifying its correctness according to the real profile. We could have presented complex and realistic topologies, but the goal of this section is to analyze in details the traffic generation process provided by our tool. By introducing additional complex topologies, the statistical analysis of the generated traffic would have been too complex given that we would have had difficulties to link a specific parameter variation with a specific cause: our tool or the complex topology. This is why we chose a simple experimental topology with only two hosts and one router. The SourcesOnOff tool has been deployed on the different hosts: the sender part on the transmitter host and the receiver part on the receiver host. For this experiment, we use two Linux Debian hosts (mono-core processor @1.2 GHz, 1GB of RAM), connected to a Linux Debian router (quad-core processor @3.8 GHz, 4 GB of RAM). The links between hosts are Ethernet RJ45 connections manually configured to 10baseTx-HD, in order to easily limit the link capacities without any software solution and thus to capture all the traffic without any loss. For each original trace replayed thanks to the SourceOnOff tool, we captured, on our experimental topology, the same quantity of generated data in order to be consistent between original and generated data. As a reminder, we present only one set of results related to data collected on 01/29/2013.

The validation of our tool needs to analyze its generated traffic. This is why this section analyzes the original captured traffic and explains how generated traffic is really close to the original network traffic. Thus, we performed different quantitative and qualitative verifications on the generated traffic to answer the following questions: does the generated traffic comply with the original one?

4.1 Statistic profile detection

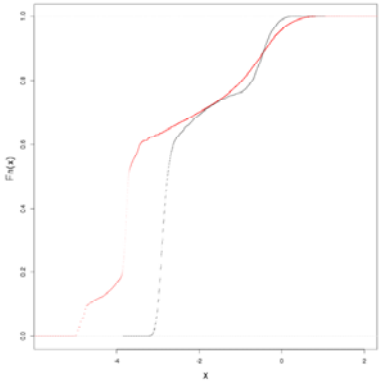

Figure 2: Fitting of original (red curve) and generated (black curve) for Doff traffic distributions

By applying the algorithm described in section 3.3, the tool is able to detect several different statistic profiles in the same original data as described in the Figure 2. We studied both Doff and Don values but, to be concise, we describe in this section only Doff (i.e. the inter-train durations) results. The Doff distribution is a complex statistical process that we have modeled based on the function composition algorithm described in the previous section. Our algorithm chose a composition between Weibull and Dirac functions (as plotted by the black curve). In the following sub section, we are going to validate how this function composition fits very well the original statistical process.

4.2 Qualitative analysis : quantile-quantile plots

As qualitative estimators, we used quantile-quantile plots diagrams (also called "QQPlots") to represent on the same diagram the measured values on the real network (i.e. the original data) and the measured values on the experimental network (i.e. the SourcesOnOff generated data). QQPlot diagrams sort independently X and Y values, and then represent points to the sorted coordinates. When the X and Y series are correlated, a linear tendency is visible on the diagram. On our QQPlot diagrams, we will represent a diagonal blue line of equation y=x, in order to represent a perfect correlation between x and y values.

The set of values used for the X data is the set of durations between two consecutive TCP connections observed in the capture in the real network. The set of values used for the Y data is the set of durations between two consecutive TCP connections measured in the traffic generated by our tool SourcesOnOff. In other words, the Y values are the durations our tool waited between starting two TCP connections.

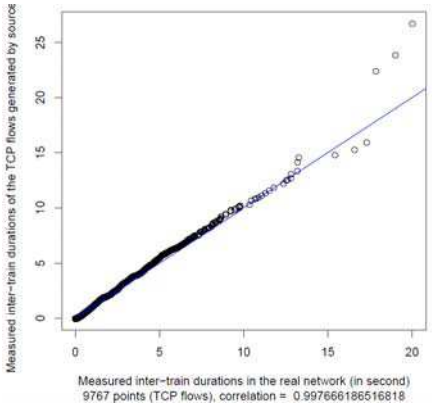

Figure 3: QQPlot for Doff values (generated traffic vs. real traffic)

We can see on figure 7 that the values are very well correlated: most of the 9,700 points drawn on this figure are near the diagonal blue line of equation y=x. This means most values measured in the real network were successfully reproduced in our experimental network. The correlation factor is 99.8% in our case for Doff values. A few points with high values are not well correlated: they come from the tail of the Weibull distribution densities.

We can conclude that our tool shows very good trends for the generated traffic compared to the original one. However, it is necessary to analyze more accurately how the generated traffic is close to original. This is why we used additional tools (BIC

distance and Hurst factor) to check quantitatively these results in the following section.

4.3 Quantitative analysis

4.3.1 BIC distance

Table 1 shows BIC distance analysis. We can note that original and generated traffics are the closest when the statistic model combines both Weibull and Dirac profiles. The qualitative similarity between original and generated traffics provided by QQPlot is thus confirmed by BIC distance computations for both Don and Doff profiles. This result validates that a combination of both Weibull and Dirac distributions is the right statistical profile to consider with our original data.

Table 1: BIC distance between original and generated traffic

Statistic model	Doff BIC distance	Don BIC distance
Weibull (+ Dirac)	**0.01558**	**0.02527**
Pareto (+ Dirac)	0.0209	0.03834
Exponential (+ Dirac)	0.08892	0.06222
Gaussian (+ Dirac)	0.1027	0.07535

4.3.2 Hurst exponent computation

We wanted also to compute quantitatively the long-term memory of the generated traffic compared to the original one. This can be showed with the Hurst exponent computation: this value is an indicator of long-term memory. In our case, we used the Wavelet-based joint estimator of the Hurst exponent, as described by D. Veitch and P. Abry in [13] to study the Long-Range Dependence. In table 2, we compare the Hurst exponent estimation for the throughput generated by our tool with the throughput of the real capture at different time scales (from 100 us to 10 s).

Conforming to [13], data (see table 2) exhibit Long Range Dependence (LRD). Indeed, data show LRD when $0.5 < H < 1$. Moreover, the dependence is stronger when H is closer to 1. Thus, we can conclude than in our case, we have a strong Long Range Dependence in both series of throughput, whatever the time window is defined. We can also conclude that both generated and original traffics exhibit the same LRD level (considering a 12 % error interval).

Table 2: Hurst exponent estimations (H)

Sample duration for throughput measurements	H for our tool SourcesOnOff	H for real throughput	Error ratio
100 us	0.88	0.83	7 %
10 ms	0.97	0.95	2 %
100 ms	1.00	1.00	0 %
1 second	1.00	0.88	12 %
10 s	1.00	1.00	0 %

5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a methodology to generate network traffic with realistic characteristics. A tool has been developed, based on the application of ON/OFF sources with different statistical profiles. Parameters of the distributions can be defined by the user or extracted from real traffic analysis. We have completed this paper with a validation of both the traffic generation methodology and the SourcesOnOff tool. Different experiments have been conducted. All of them appear to validate the proposition that our tool is able to generate traffic with the same characteristics as real ones.

The tool is freely available and may be utilized for a wide variety of network traffic profiles. We hope users can appreciate the wide range of applications where SourceOnOff can be utilized. This tool is currently unable to generate traffic other than TCP and UDP protocols. In the future, the tool may be developed further in order to support ICMP protocol. Moreover, the tool may support other statistical distribution profiles and may provide additional statistics for the users. Having completed the validation of this tool, we can now take into account more complex network topologies (cloud computing applications for instance) and distribute different SourcesOnOff sender and receiver agents among them for the future experiments we plan to conduct.

6. REFERENCES

[1] Olivier P. and Benameur N., Flow Level IP traffic characterization, France Télécom, 2000

[2] Gebert S., Pries R., Schlosser D. and Heck K. 2011 Internet Access Traffic Measurement and Analysis, Univ of Würzburg, Inst. Of Computer Sciences, Germany, Hotzone GmbH

[3] Leland W. E., Taqqu S. M., Willinger W. and Wilson D. V., On the Self-Similar Nature of Ethernet Traffic, (Extended Version) IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 2, NO. 1, FEBRUARY 1994

[4] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/, 2013/01/21

[5] Wireshark, Go Deep, http://www.wireshark.org/, 2013/01/21

[6] iperf, a modern alternative for measuring maximum TCP and UDP bandwidth performance, http://iperf.sourceforge.net/,2013/01/21

[7] BWPing, Open Source bandwidth measurement tool, http://bwping.sourceforge.net/, 2013/01/21

[8] The NetPerf HomePage, http://www.netperf.org/netperf/, 2013/01/21

[9] The R Project for Statistical Computing, http://www.r-project.org/, 2013/01/21

[10] Tcpdump & LibPcap public repository, www.tcpdump.org, 2013/01/31

[11] IPsumDump and IPaggCreate website, http://www.read.seas.harvard.edu/~kohler/ipsumdump/, 2013/01/31

[12] Schwarz, Gideon E, Estimating the dimension of a model, Annals of Statistics 6 (2): 461–464, 1978

[13] Veitch T. and Abry P. 1999 A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence, pp.878-897, Vol.45(3), IEEE Trans. Info. Theory, special issue "Multiscale Statistical Signal Analysis and its Applications"