Design of Software Network Traffic Generator

PETR ZACH, MARTIN POKORNY, ARNOST MOTYCKA
Department of Informatics
Mendel University in Brno
Zemedelska 1, 61300 Brno
CZECH REPUBLIC
petr.zach@mendelu.cz, martin.pokorny@mendelu.cz, mot@mendelu.cz

Abstract: - This contribution focuses on the issue of the network traffic generation tools. Firstly, current state of the existing network traffic generators is considered. Based on the state of currentnetwork traffic generators and based on own requirements on the network traffic generatora core of new traffic network generator is proposed. After that, usage of the developed network traffic generator is presented on an example where QoS parameters of a voice traffic floware examined.

Key-Words: -network traffic generator, realistic network traffic, computer network, end-user emulation, network traffic capturing, Java

1 Introduction

Development of computer network technologies and growth of Internet services change meaning and usage of the Internet continuously. The whole system consists of a broad spectrum of heterogeneous services and technologies with different operation requirements. With increased demand for real-time multimedia services, several types of traffic flows traverse the network infrastructure, thus a QoS (Quality of Service) testing is required. Moreover, number of the Internet users grows continually, thus security assurance is one of primary goals of current private networks. Network traffic generators help network administrators, developers and researchers to prepare, validate and install technologies ensuring secure and properly working network infrastructure.

Traffic generators can be classified according to several criteria but essentiallytraffic generatorsare either hardware devices or software tools. Hardware generators (e.g. Ixia IxChariot) usually achieve higher performance and accuracy in comparison with software tools whose performance is dependent on many circumstances (end-device performance, operating system, etc.). On the other hand, hardware devices are usually commercial products, whereas software tools are generally open source or cost-effective tools developed by researchers and enthusiasts. Despite these arguments, software network traffic generators are widely used in networking area due to economics aspects and their flexibility and customization [1].

Several network experiments are performed in the Laboratory of computer networking at the Department of Informatics (Faculty of Business and Economics, Mendel University in Brno) [2].The network traffic generator is employed by students and researchers. Couple of network generators was examined during the time but none of them fully meets our requirements (see below). A group of university staff and students participate on the development of a new solution that can extend the set of network traffic generators used in research as well as in production networks.

Contribution of this article is to propose a core of our own software network traffic generator (NTG), the extent of this article covers main features and architecture of NTG.First, requirements on network traffic generators are specified. After that, a comparison of selected existing network generators is presented (in section 2). According to specified requirements and features of existing tools, new solution is proposed and basic network experiment is described (in section 3). Conclusion follows in section 4.

2 Problem Formulation

Generation of network traffic is fundamental for several networking research areas as mentioned above, for example performance of networks and network devices [3], security (e.g. firewalls [4], intrusion and anomaly detection, background and malicious workload), quality of service and quality of experience, new protocols, frameworks [5] or

proposal verification [6,7], available bandwidth measurement in real networks, etc. [1].

Regardless of a specific use, the main goal of the network generator is to produce various types and characteristics of traffic that emulate realistic behavior of large amount end-users. Many articles deal with network traffic simulation, analysis and modeling. Authors in [8] propose a solution for traffic simulation in ns-2 network simulator, [9] deals with traffic modeling, authors in [10, 11] propose a solution for realistic HTTP traffic generation. Articles in [12, 13, 14] analyze the production network traffic behavior.

According to authors in [15], network traffic can be generated in three ways:

- stochastic generation,
- replication of production network traffic,
- using list of instructions (communication scenario) for applications in the tested network.

Authors in [16] divide network generators according to the layer on which they work:

- Application-level traffic generators: They emulate the behavior of specific network applications in terms of the traffic they produce.
- Flow-level traffic generators: They are used when the replication of a realistic traffic is requested only at the flow level (e.g., number of packets and bytes transferred, flow duration). For example, Bit-Twist [19] represents this group.
- Packet-level traffic generators: With this term we refer to generators based on packet's Inter departure time (IDT) and packet size (PS). The size of each packet sent, as well as the time elapsed between subsequent packets, are chosen by the user, typically by setting a statistical distribution for both variables. Most of current packet generators belong to this group.

Authors in [8, 10, 16, 17], recognize two approaches of network traffic generation: open-loop and closed-loop. In the closed-loop mode, there is a fixed number of users using the network and their services. Each of these N users repeats these two steps, indefinitely: (i) submit a job, (ii) receive the response and then wait for some time before a next job is submitted ("think" time). In the closed system, a new request is triggered by the completion of a previous requestonly. In the open system model there is a stream of arriving requests with average arrival rate. The differentiating feature of an open system is that a request completion does not trigger a new request: a new request is only triggered by a new request arrival [17]. A hybrid approach, discussed in [10], combines aspects of the closed- and open-loop approaches. With a hybrid approach, user sessions are initiated at specified time. This is similar to the open-loop approach in that a new session can be initiated before the previous sessions are finished. However, similar to the closed-loop approach, within each session a request can be issued only after the response to the previous request in that session has been received. Authors in [17] claim that the hybrid mode is more representative of real systems than closed- or open-loop approach.

2.1 Comparison of open-source software traffic generators

Several software-based network traffic generators were reviewed.Following text compares selected five free network traffic generators in alphabetical order. These generators were chosen with intention to cover application level, flow-level as well as packet-level traffic generators. Detailed list of alternative tools is summarized in [18]. The comparison (see Tab. 1) takes into account five criteria. The desired tool should be platform-independent (PI), distributed (D), no proprietary receiver is needed (R), generation of the multimedia traffic (RTP) has to be supported, data analysis (A) and statistics have to be present.

Table 1: Comparison of selected open-source software traffic generators.

Name	PI	D	R	RTP	A
Desired tool	Yes	Yes	No	Yes	Yes
Bit-twist	Yes	No	No	Yes	No
D-ITG	Yes	Yes	Yes	Yes	Yes
Karat	No	No	No	No	Yes
Ostinato	Yes	Yes	No	No	Yes
Scapy	No	No	No	No	Yes

Bit-Twist [19] is a libpcap-based Ethernet packet generator that replicates traffic from captured PCAP files [20]. D-ITG [21] is one of leading application and packet-level network generators. This distributed andplatform-independent tool enables many features and has great future potential.The architecture of the NTG is inspired by the D-ITG in several ways. Scapy [22] as well as Karat Packet Builder [23] are packet-level traffic generators particularly suitable for firewall testing using variety of network protocols. Packet-level traffic generator

ISBN: 978-960-474-341-4

Ostinato [24] treats the user with a sophisticated GUI and detailed packet header customization.

2.2 User requirements on proposed solution

Regardless of characteristics and features of the reviewed open source tools we tried to specify user requirements on the network traffic generator (NTG) we would like to use for network experiments in our network laboratory.

2.2.1 Brief description of NTG

The primary purpose of this tool is to emulate network end-users by a real network traffic generation. Besides essential network traffic generation, data collection and analysis as well as advanced user interface should be present in the end.

The data analysis contains calculation of severalmeasured parameters/statistics described e.g. in [25]: number of task failures, mean time between failures (MTBF), bitrate, download speed, one-way delay, round-trip time (RTT), jitter, packet loss, etc.

NTG should be designed asplatform-independent, distributed and centrally controlled software running on network end-devices in the form of an application or a service. NTG should behave as real end-users initiating regular client requests and exchanging real data with native server services. Some of the listed use cases (performance testing, quality of service testing) implythe emulationof a large amount of end-users. A communication scenario (CS) is necessary to schedule the sequence of particular tasks. The use of the tool has to betransparent and intuitive considering implementation of all required functions, accurate traffic generation and ability of future development.

2.2.2 Functional requirements

To meet requirements from previous paragraph main NTG features (Fig. 1) are listed:

- Communication scenario management.
- Management of remote entities generating network traffic (senders) on particular end-devices.
- Network traffic generation from particular senders.
- Collecting data about ongoing test.
- Statistics reporting based on collected data.
- User interface (including management of the whole test and remote components).

2.2.3 Operational and limiting requirements

Because of the end-user emulation, the NTG assumes presence of properly configured network services, for example a web server. If a user using an end-device A accesses a web page on an non-responding server B, communication will fail. NTG behaves in the same manner as the end-user that means it treats the server as an unreachable one.

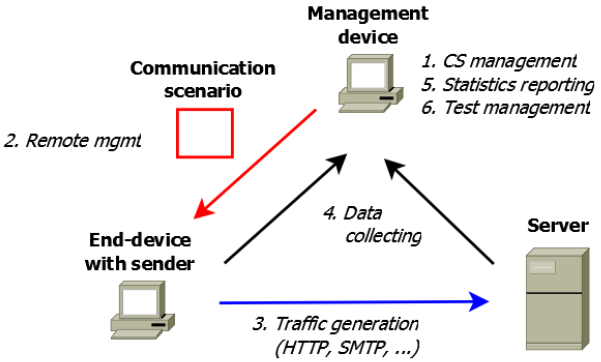

Fig. 1: Schema of the main functions.

3 Problem Solution

Despite the fact that many tools for the traffic generation are available, none of the reviewed tools fully suits all the requirements on network traffic generator required for network experiments and education at the Mendel University in Brno. Proposed NTG is proposed according to our requirements and pros and cons of other described solutions as well. NTG is considered as application-level traffic generator working in hybrid-close mode (described in 3.3).

3.1 Design of NTG components and their relationships

NTG has to be a distributed system spread across tested network infrastructure to generate traffic and analyze data about tested network accordingly.The NTG's complexity (traffic generation, data collection, statistics) requires heterogeneous components, while the whole system should be easily manageable from a single host. Considering these facts, four types of componentswere specified: Control component (M - management), Sender (S, particular traffic generator), Capture (C), Statistics (ST). Component schema and relationships between them shows Fig. 2.

Essential component of whole system is sender that represents end-user by generating network traffic. The S can occur more than oncein tested topology (described in subsection 3.2).

ISBN: 978-960-474-341-4

TheNTG initiates correct communication sessions and exchanges data with native server services (Fig. 2, black dashed arrow). Currently, ICMP, HTTP, POP3, SMTP and RTP protocols are supported. The set of supported protocols is easily extensible.

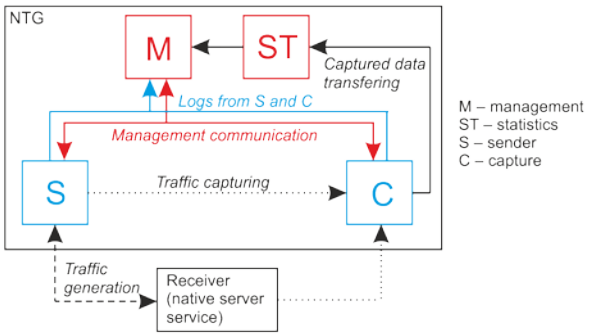

Fig. 2: Component schema.

The component C acts as a capture of the data that travels through the tested topology. The C can occur more than once in the tested topology and the user of NTG can decide at which places the desired data about traffic should be collected in the tested network.

Components S and C are driven by the management (control) component. The component M is NTG's head point, which is represented by the GUI where user is able to manage the entire toolcentrally. The M delivers the particular communication scenarioto the S component. The scenario contains instructionswhen where, and what has to be generated. In the same manner, the C receives instructions from the M about how long and which type of traffic has to capture.

The Data collected by C components are transferred to the ST after the test. The ST extracts the data from captured PCAP files received from all Cs and calculates measured parameters. After processing the data, the ST supplies statistics to the M where user can view/export the desired report.

Multiple types of control communication between components are required: (i) management (Fig. 2, red arrow), (ii) logging (Fig. 2, blue arrow), (iii) captured data transfer (Fig. 2, black arrow). Management control communication is used by the M to control the remote components S and C. The logging control communication acts in the opposite direction (from S/C to the M) in comparison with the management traffic and is used to inform the M about S/C behavior. In case of the S, the logs also contain information about result of a particular task execution. In case of C, the information about success or failure of the packet capture is part of the logs as well. Consequently, the user of the NTG is informed about testing process immediately. After the test, captured data is transferred from each participating C component to the ST. It should be also possible to transmit captured data during the test in smaller quantity due to limited memory space for storing captured data on the end-device running the particular C.

According to previous paragraphs, the network testing process using this tool can be depicted as simplified process diagram (see Fig. 3).

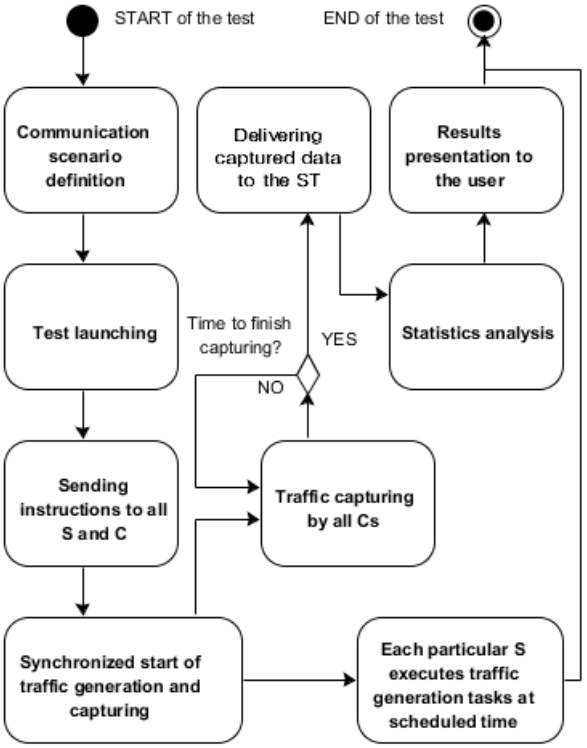

Fig. 3: Testing process using NTG.

3.2 Component deployment in the tested topology

Due to the distributed system architecture and required control communication, there is a question how to avoid interference with an ongoing test by the control communication. There is a risk that the generated traffic completely consumes all the available bandwidth and the remote components may be temporarily unreachable. Management control communication is delivered to the S and C components before the own test but logs as well as PCAP files can traverse the infrastructure during the test phase. Each logged message is represented by one UDP packet, their frequency depends on scenario content. Size of individual PCAP files reach megabytes. Each PCAP file is transferred to the ST component once its threshold size is reached.Essentially, two problems must be resolved

ISBN: 978-960-474-341-4

in relation to the control communication in the tested network: (i) influencing the test results, (ii) risk of temporary remote components unreachability caused by link saturation. This problem can be solved in two ways: (i) using the tested network for both the generated and control traffic, (ii) using dedicated infrastructure for the control traffic only.

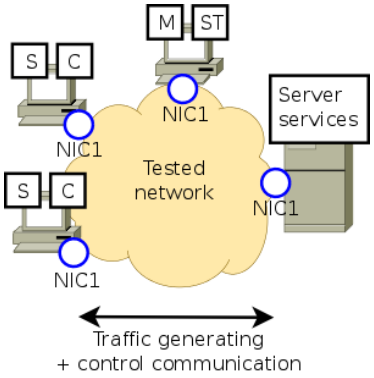

Fig. 4:Deployment of NTG components with control communication in tested network.

Transferring the control communication inside the tested infrastructure (Fig. 4) is easier to set up theexperiment but measured parameters can be distorted in some cases. For exampleQoS test results (delay, jitter, packet loss) can be affected by control communication presence. To avoid this issue, a dedicated management network has to be involved, as depicted on Fig. 5. Using this approach, the control communication is completely isolated from tested network, thus no measured parameters can be affected in the tested topology. On the contrast, two disadvantages are associated with this approach – two network interfaces are required on all active end-devices in the tested topology and it takes longer to prepare experiment. Both ways offer some pros and cons, the choice depends on a nature of the test. Anyway, NTG provides both of them.

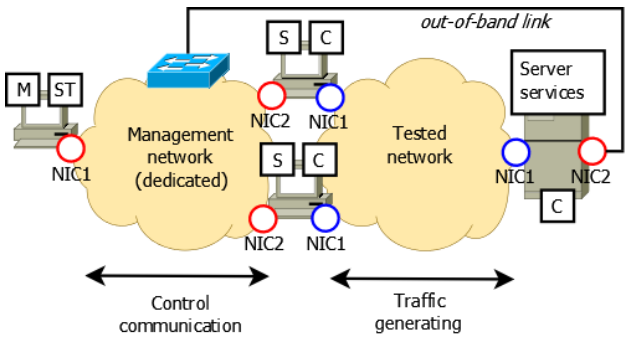

Fig. 5: Deployment of NTG components with control communication in dedicated network.

Considering these issues, it is necessary to define required number of particular components. The NTG contains always only one M and one ST. Multiple instances of the S and C components can be placed in the tested topology as mentioned above. Generally, the S has to be on each end-device dedicated to the traffic generation, and the C has to be on each end-device where traffic capture is required. There are two solutions how to place the C component on the end-devices in the tested topology: (i) Deploy the Cs on all end-devices with the S components only (Fig. 4) to gain view about the generated traffic on each S. Using this, there is no information about measured parameters on the receiving device. Anyway, some parameters such as Round Trip Time can be discovered in this way. (ii) Deploythe Cs on each end-device that acts in the tested topology (Fig. 5) to be able to gain measured parameters from both the sending and the receiving device. This is necessary for one-way delay or packet loss of UDP flow calculation.

Each particular end-device can host only one instance of each component type, although all four NTG components (M, ST, S and C) can be presenton one end-device. Required numbers of components are described in (1) where $S_m$represents the set of S components (similarly toM_m, ST_m, C_m).

$$\begin{aligned} |A_m| &= |ST_m| = 1 \\ |S_m| &\geq 1 \\ |S_m| &\leq |C_m| \end{aligned} \tag{1}$$

3.3 Communication scenario

Traffic generation parameters are stored in the communication scenario. NTG enables to construct the communication scenario as a “long story” about all events that occur in the tested infrastructure during the test. Each particular record contains instructions what the S should generate on end-device. The record is represented by a set of these attributes: sender ID, execution time, number of repetitions, wait (“thing”) time between two repetitions, selected protocol and protocol details. Protocol attributes differ according to selected protocol. Once the scenario is completed, it is delivered to all the active S components and each S receives only its relevant part of the scenario, as depicted on Fig. 6.Considering closed-, open-loop- or hybrid mode, NTG acts as hybrid mode network traffic generator.

ISBN: 978-960-474-341-4

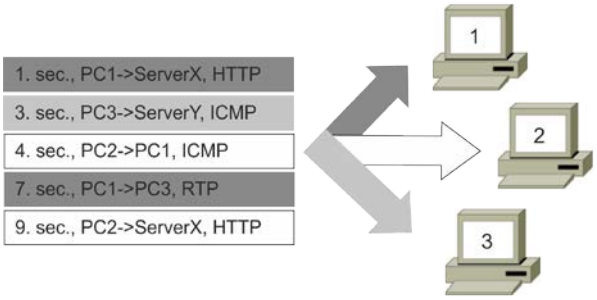

Fig. 6: Communication scenario concept.

NTG as a distributed tool is highly dependent on precise synchronized time in tested infrastructure. To provide traffic generation in scheduled time and capture traffic with proper timestamps equal time on all end-devices is a must. NTG relies on preceding time synchronization, for example using NTP (Network Time Protocol).

3.4 NTG core implementation

NTG was developed in Java programming language that provides platform-independent usability. Currently, the core (back-end) of NTG is implemented including complete functionality of the components S and C. Component M is controlled from CLI. There is an API available for the GUI (front-end) implementation, which has to contain its own scripting language for advanced communication scenario management. Component ST communicates with the others components but its whole functionality is matter of future work.

Multithreading is incorporated in the NTG software design. Each task of the communication scenario on the component S is maintained by separate thread to accurately follow predefined scenario and mitigate task execution delays.

Third-party software and frameworks are employed. Especially, the tcpdump[20] is embedded in the C componentto capture packets.

3.5 Example of NTG usage

Object of this experiment is to analyze the impact of insufficient bandwidth (BW) on voice streaming and HTTP flows.There are one RTP flow (UDP) and several HTTP flows (TCP) traversing the topology as defined in the communication scenario. The whole test was performed twice – in the first case, network bandwidth is unlimited (respectively to 1Gigabit). In the second case, the bandwidth is limited to 1 Mbps in one direction (full duplex). Bitrate of all flows is analyzed at the receiving side as well as delay, jitter and packet loss of the UDP flow and download time of all individualHTTP downloads.

Network topology (see Fig. 7) consists of tested and management network. PCs are running Fedora 16 64bit, servers CentOS 6.2 64bit operating systems. Both of theL3 switchesare Cisco Catalyst 3750 (WS-C3750-24P) with IOS version 12.2(44)SE5, IP Services and 131072 KB RAM. The link between interfaces Fa1/0/1 is intentionally policed to 1 Mbps in both directions. IP addresses are configured in a static fashion.The NTG is used for traffic generation as well as for data collection, the analysis is processed using Wireshark [26].

One test takes 100 seconds. The test starts at time 0. Inthe 1st second PC1 initiates a request to download 10MB file in order to the WWW server. In the 3th secondthe PC2 starts receiving RTP traffic (voice) from RTP server. RTP traffic is represented bya 55 seconds long voice record with CBR 320 kbps. In the 6th second PC1 starts downloading 0.5MB file from the WWW server. This is repeated five times with wait (“think”) time of 3 seconds between the requests.

NTG's components are arranged as depicted on Fig. 7. The Cs are located on each active host, while the S componentsare present only on the RTP server (sends RTP packets) and on PC2 (initiates file downloads from the WWW server). The M and ST are located in the management network. The C and S components communicate with the M and ST through the management network, thus no measured parameter can be affected by thecontrol communication in the tested network.

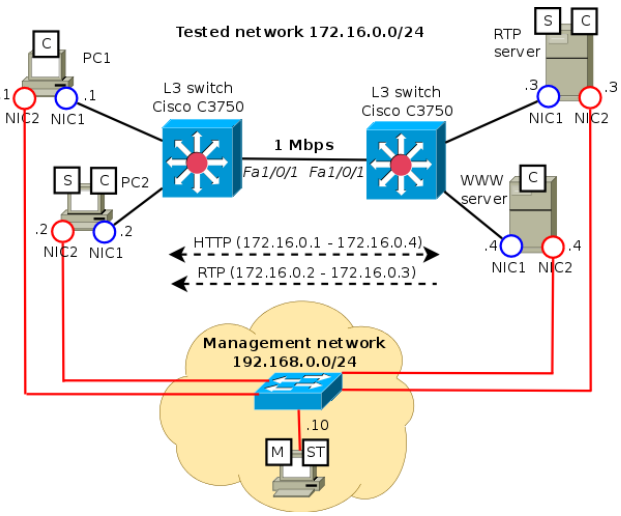

Fig. 7: Experiment topology diagram.

In case of unlimited bandwidth, bitrate of all traffic traverses the inter-switch link in theright-to-left direction. Fig. 8shows traffic with symmetrical peaks of each downloaded file. No RTP packetwas

ISBN: 978-960-474-341-4

dropped, as summarized in Table 3. In case of BW limitation, the received bitrate, as depicted on Fig. 9, is noticeably limited by the available bandwidth of 1 Mbps. The RTP flow was significantly disturbed by the present TCP flows (Fig. 11), which results in several packet drops (Table 3). Download of particular files from the HTTP server took significantly longer time (Table 2).

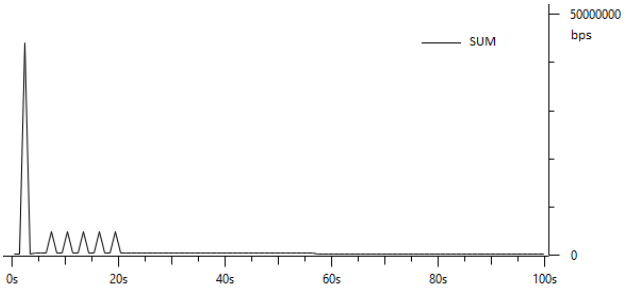

Fig. 8: Bitrate of all traffic traversing the inter-switch link in the right-to-left direction (without BW limitation).

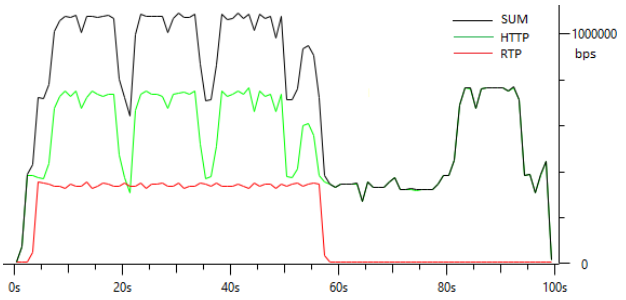

Fig. 9: Bitrate of all traffic traversing the inter-switch link in the right-to-left direction (with BW limitation).

The impact of limited bandwidth on RTP flow is evidenton Fig. 10 and Fig. 11. In case of unlimited bandwidth, the RTP flow is constant, while the bitrate irregularity is obvious in case of bandwidth limitation.

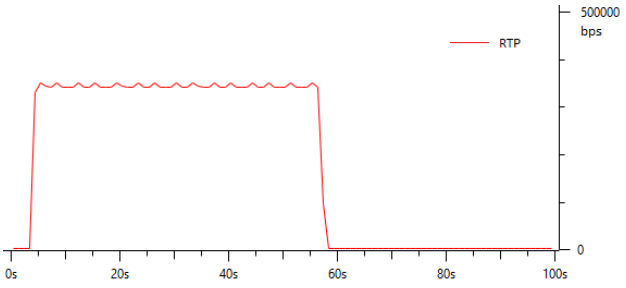

Fig. 10: Received bitrate of RTP flow on PC2 (without BW limitation).

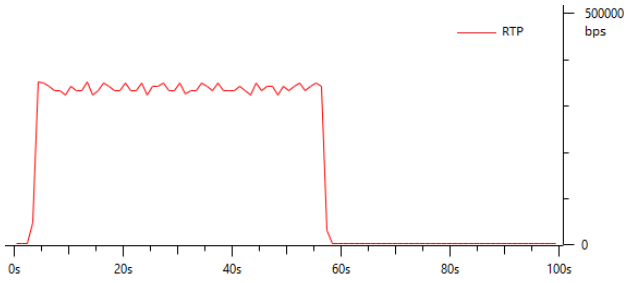

Fig. 11: Received bitrate of RTP flow on PC2 (with BW limitation).

The transmission quality decreased distinctly due to the limited bandwidth consumption. The traffic exceeding permitted limit is dropped, regardless of packet type (TCP, UDP). Consequently, the quality of voice transmission is decreased by higher packet loss and the download of individual files from the WWW server takes longer because of packet retransmission. The download time is summarized in Table 2, packet loss ofthe RTP flow is shown in Table 3. Mean delay and jitter of the RTP flow is minimal (6.02 ms, resp. 0.58 ms) in both of the tests because of queuing absence (exceeded traffic is dropped).

Table 2: Download time of individual files.

	Without BW limit [s]	With BW limit [s]
10MB file	0.43	87.01
500kB file 1	0.46	12.39
500kB file 2	0.47	9.38
500kB file 3	0.46	12.18
500kB file 4	0.46	26.07
500kB file 5	0.45	11.98
Mean value (1–5)	0.46	14.40
Std. dev. (1–5)	0.01	6.64

Table 3: Packet loss of RTP flow.

	Delivered packets	Packet Loss	
		Count	%
Without limit	2036	0	0.0
With limit	2000	36	1.8

4 Conclusion

This contribution discusses the issue of tools for network traffic generation. Design of a new software network traffic generator was proposed, at the same time features of current solutions were considered to create complement of existing tools. In the end, an illustrative network experiment with the NTG was described that shows basic usability of this tool.

ISBN: 978-960-474-341-4

The NTG assists researchers in experiments across the networking research field. Benefits of NTG deployment in academic as well as in commercial area are obvious. Quality of Service assurance or security risk mitigation has positive economic consequences in the end.

Our future work focuseson completion of remaining parts followed by performance and accuracy testing. Above that, in todays wireless and cellular phone networks expansion NTG should be able to cooperate with smart phones as well.

Acknowledgments:
This work has been supported by the research project IGA 3/2013.

References:

[1] Botta, A., Dainotti, A., Pescape, A., Do you trust your software-based traffic generator?, *Communications Magazine, IEEE* , Vol.48, No.9, 2010, pp.158-165. ISSN 0163-6804.

[2] Pokorný, M., Zach, P., Design, implementation and security of a typical educational laboratory computer network. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*. Vol. 51, No. 4, 2013, pp. 1077-1087. ISSN 1211-8516.

[3] Bradner, S., McQuaid, J., RFC 2544: Benchmarking Methodology for Network Interconnect Devices, *IETF*, 1999.

[4] Hickman, B., Newman, D., Tadjudin, S., Martin, T., RFC 3511: Benchmarking methodology for firewall performance, *IETF*, 2003.

[5] Anitha, P., Chandrasekar, C., A Framework for Configuration and Management Of Quality-Of-Service (Qos) in Wireless Zigbee Networks, *WSEAS Transactions on Communications*, Vol. 11, No. 4, 2012, pp. 147-157.

[6] Liang, Ch., Pan, S., Wu, J., A Sustained QoS Solution by Contention Adaptation in IEEE 802.11e Wireless LANs, *WSEAS Transactions on Communications*,Vol. 10, No. 10, 2011, pp. 287-296.

[7] Gomathi, N., Seethalakshmi, P., Govardhan, A., An Integrated Cross Layer Approach for Multimedia Streaming using Multipath and Multiple Description Coding in Mobile Ad-Hoc Networks, *WSEAS Transactions on Communications*,Vol. 11, No. 2, 2012, pp. 57-69.

[8] Weigle, M., et al., Tmix: a tool for generating realistic TCP application workloads in ns- 2, *ACM SIGCOMM Computer Communication Review*, Vol. 36, No. 3, 2006, pp. 67–76.

[9] Barford, P., Crovella, M., Generating representative Web workloads for network and server performance evaluation, *SIGMETRICS – Performance Evaluation*, Vol. 26, No. 1998, pp. 151–160.

[10] Hashemian, R., Krishnamurthy, D., Arlitt, M., Web workload generation challenges – an empirical investigation, *Software: Practice and Experience*, Vol. 42, No. 5, 2011, pp. 629-647.

[11] Kolesnikov, A., et al., Web workload generation according to the UniLoG approach, *OASICS*, Vol. 17, No. 1, 2011, pp. 49-60.

[12] Choi, Y., Silvester, J., Kim, H., Analyzing and modeling workload characteristics in a multiservice IP network, *IEEE Internet Computing*, Vol. 15, No. 2, 2010, pp. 35–42.

[13] Abhari, A., Soraya, M., Workload generation for YouTube, *Multimedia Tools Applications Journal*, Vol. 46, No. 1, 2010, pp. 91–118.

[14] Veloso, E., et al., A hierarchical characterization of a live streaming media workload, *IEEE/ACM Transactions on Networking*, Vol. 14, No. 1, 2006, pp. 133–146.

[15] Vishwanath, K., Realistic and responsive network traffic generation. *ACM SIGCOMM 2006*, Vol. 36, No. 4, 2006, pp. 111–122.

[16] Botta, A., Dainotti, A., Pescape, A., A tool for the generation of realistic network workload for emerging networking scenarios, *Computer Networks*, Vol. 56, No. 15, 2012, pp. 3531-3547.

[17] Schroeder, B., Wierman, A., Harchol-Balter, M., Open versus closed: A cautionary tale. *Networked Systems Design & Implementation*, Vol. 3, No. 1, 2006, pp. 239-252.

[18] Traffic Generators for Internet Traffic, http://www.icir.org/models/trafficgenerators.html

[19] Bit-Twist. Libpcap-based Ethernet packet generator, http://bittwist.sourceforge.net/

[20] TCPDUMP/LIBPCAP public repository, http://www.tcpdump.org/

[21] DIT-G, Distributed Internet Traffic Generator, http://traffic.comics.unina.it/software/ITG/documentation.php

[22] Scapy, http://www.secdev.org/projects/scapy/

[23] Karat Packet Builder, https://sites.google.com/site/catkaratpacketbuilder/

[24] Ostinato, https://code.google.com/p/ostinato/

[25] Evans, J., Filsfils, C., *Deploying IP and MPLS QoS for Multiservice Networks: Theory & Practice*, Elsevier, 2007.

[26] Wireshark, http://www.wireshark.org/