Network Traffic Generation: A Survey and Methodology

OLUWAMAYOWA ADE ADELEKE, NICHOLAS BASTIN, and DENIZ GURKAN,
University of Houston

Network traffic workloads are widely utilized in applied research to verify correctness and to measure the impact of novel algorithms, protocols, and network functions. We provide a comprehensive survey of traffic generators referenced by researchers over the last 13 years, providing in-depth classification of the functional behaviors of the most frequently cited generators. These classifications are then used as a critical component of a methodology presented to aid in the selection of generators derived from the workload requirements of future research.

CCS Concepts: • **Networks** → **Network experimentation**; **Network performance analysis**; **Network measurement;**

Additional Key Words and Phrases: Network, packet, traffic, workload, generator, experiment, survey, analysis

ACM Reference format:
Oluwamayowa Ade Adeleke, Nicholas Bastin, and Deniz Gurkan. 2022. Network Traffic Generation: A Survey and Methodology. *ACM Comput. Surv.* 55, 2, Article 28 (January 2022), 23 pages.
https://doi.org/10.1145/3488375

1 INTRODUCTION

The internet has become ubiquitous. Although it started as a small network with wired connections between 4 computers in 4 universities in the western part of the USA, it has evolved into a massive web with over 18 billion networked devices and over 3.9 billion users as of 2018, according to the Cisco Annual Internet Report [39]. The implication is that nearly half of the population of the world uses internet based services on a daily basis, and numbers continue to increase every day. This sustained increase in the internet size and utility continues to ride on the tireless work of researchers in the field of computer networks and distributed computing.

Over the last decade, there has been significant research output from academia in the field of computer networks as a result of the advent of **software defined networking** (**SDN**) and **network function virtualization** (**NFV**). In-depth research experiments with network topologies that resemble production networks in terms of the number and diversity of nodes and links have become considerably more accessible. Consequently, more sophisticated requirements arise on traffic workloads in order to provide a realistic testing environment. Since actual production traffic

This work is funded in part by the National Science Foundation (NSF) Division of Computer and Network Systems (CNS) core grant award no. 1908974.
Authors' address: O. A. Adeleke, N. Bastin, and D. Gurkan, University of Houston, 4730 Calhoun Rd, Houston, Texas, 77204; emails: {oaadeleke, nbastin, dgurkan}@uh.edu.

0360-0300/2022/01-ART28 $15.00
https://doi.org/10.1145/3488375

traces are almost never available to academic researchers due to privacy policies [127], a plethora of traffic generators are utilized in network science and engineering research. Even when privacy policies are not an issue [53, 95, 124], the logistical hurdles of scaling an existing production traffic capture into a testbed can be daunting. Furthermore, the effective replay of existing traffic traces only enables very specific research experiments where the local topology matches the exact graph of the network where the packets were originally captured. In essence, the capability to replay such traces becomes considerably limited on network topologies that differ significantly from the original capture network. Therefore, researchers have to resort to alternative methods for creating traffic workloads for their experiments, and one of the most popular options is the use of traffic generators. This article presents a survey on traffic generation methods and a selection methodology for traffic generators to match experiment objectives in applied research.

We present a comprehensive survey of network traffic generators in academia and industry. Unlike existing traffic generator surveys [48, 86, 101], our objective is not a performance comparison, rather a determination of the functional behaviors. The performance of traffic generators has been studied extensively in the literature, and our survey focuses on the types of variances and functionality of the available traffic generators even though it is possible that they could be run in a high-performance setting with the support of hardware platforms and techniques guaranteeing a wire-speed generation capability. In fact, most generators in our analysis are software programs that are vulnerable to the limitations of the runtime environment and the hardware systems. Our goal is to analyze available characteristics and features of commonly preferred tools, provide a structured digest of our compilations on these features, and then to present a systematic methodology to pick suitable generators for individualized research goals.

We first present our survey of traffic generators and their usage in a comprehensive set of publications in the top ACM and USENIX conferences where we collect information on the usage frequency of a traffic generation method of any kind. (IEEE publications were not included in our corpus as the API to pull papers from the IEEE database made it difficult to perform extensive downloads of a large number of papers.) We compile over 90 traffic generators used in academia and industry. For each one, we attempt to obtain the binaries or the source code and then study available documentation or reference papers. We then categorize the generators into a taxonomy based on what kind of traffic they are able to generate. Afterward, based on the usage scenarios in papers from prestigious networking research conferences over the last 13 years (over 7,000 papers), we rank them per popularity using our custom built analysis tool [105]. The top 10 of these generators are analyzed in more detail for their individual features.

The article is organized to cover the analysis of the survey results first in Section 2. The next section is on the classification of traffic generation tools. We then present the top 10 popular traffic generators, their features, and a process for traffic generator selection in Section 4. In Section 5, we provide an overview of existing surveys in the literature and finally, we conclude in Section 6.

2 SURVEY OF TRAFFIC GENERATORS AND THEIR USAGE IN RESEARCH

In this section, we provide a comprehensive list of traffic generators (Section 2.1), and we provide results of analysis of tool popularity in Section 2.2. We assembled an exhaustive list of network traffic generators used across research and industry, finding 92 traffic generators created between 1995 and 2018. Our list of traffic generators was sourced from computer networking research papers (over 7,000 papers in [120, 121]) and general internet document searches [90, 108, 126, 135].

2.1 List of Traffic Generators

The Table 1 below lists all 92 traffic generators we considered in this survey. We have included the information on licensing, software maintenance status, supported operating systems, the generation category as outlined in the taxonomy Section 3, and the best available web link to get further information about each traffic generator. The generators in the table have been listed in descending order of popularity based on our findings in the Section 2.2.

2.2 Tool Popularity

We collected the 92 traffic generators listed in the Table 1 based on their usage in papers published over a 13 year period (from 2006 to 2018). We started at 2006 to capture traffic generators usage trends beginning from the early days of virtualization and software defined networking. Using our custom built analysis tool [105] we examined a total of 7,479 computer networking related papers, including 2,856 papers published in various conferences and journals by the Association for Computing Machinery's (ACM) Special Interest Group on Data Communications (SIGCOMM) [120]. The ACM conferences we explored include ACM-ICN [64], AllThingsCellular [66], ANRW [69], APNet [72], CHANTS [57], Cnet [41], GreenNets [63], HomeNets [61], HotNets [62], HotSDN [65], IoTS&P [70], LANCOMM [68], MECOMM [71], MobiArch [54], NetAI [56], NetEcon [60], NSEthics [67], NSDR [59], SIGCOMM [58], SOSR [55], and 43 others. The remaining 4,623 papers were published in various conference proceedings and journals of the Advanced Computing Systems Association (USENIX) computer networking related conferences [121] between the years 2006 to 2018. The conferences we explored include the ATC [10], APSys [20], CoolDC [24], CSET [17], EVT [9], FOCI [22], HotCloud [19], HotEdge [25], HotSec [7], IPTPS [18], LISA [8], NetDB [14], NSDI [12], ONS [23], OSDI [13], Security [11], SRUTI [6], SustainIT [6], SysML [16], WASL [15], WebApps [21], and 37 others. We could not include any of the Institute of Electrical and Electronics Engineers (IEEE) papers in the analysis because the API of the IEEE digital library made it difficult to perform extensive downloads of a large number of papers.

We conducted a detailed n-gram analysis of all 7,000+ papers. First, we created a list of terms/phrases that uniquely describe each identified generator, collected from a sample of referenced papers. For example, search terms for the DPDK packet generator included "dpdk pktgen", "pktgen dpdk", "dpdk packet generator", "dpdk generator", "dpdk based packet generator", and "dpdk based generator". We then created n-gram indices with n = 1 to 5, from the raw text of the entire corpus of selected papers. We searched these indices to locate matches of the traffic generator terms/phrases across the entire set of papers. For each match, we ran a script that captured the surrounding sentences for the location of the match, which resulted in about 1,800+ papers. We manually examined the sentences for each match in order to determine whether the generator was actually used, cited, or just merely mentioned in the article. Based on the surrounding text we were also able to identify and exclude cases where the search terms in the article was found to refer to something other than the traffic generation context. The scripts that we wrote and used for the search and analysis of papers is open source and made publicly available online [105].

The result of the analysis is a set of traffic generators and the associated lists of papers where the generators are used, cited, and mentioned. Based on the result, we rank the generators and select the top 10 based on their usage popularity in the last 13 years for further examination. The top 10 list consists of: `iperf2` [99], `netperf` [78], `httperf` [102], `moongen` [47], `scapy` [33], `linux pktgen` [109], `netcat` [76], `TCPreplay` [5], `iperf3` [91], and `DPDK pktgen` [149] in descending order of usage. Figure 1 gives the details of the results of this analysis, and we recall that Table 1 gives the complete list of traffic generators in descending order of usage. All top 10 traffic generators are open source [111], and they are all software-based generators.

The usage reference of each of these generators is given in Figure 2 per year from 2006 to 2018. Based on the results, constant/max throughput traffic generators—particularly `iperf2` [138]—continue to dominate in terms of usage. More recently developed realistic traffic generators that are based on stochastic models, e.g., `swing`, `DITG`, and so on, are not cited as frequently as the

Table 1. Status of Traffic Generators in Research and Industry as of Jan 2021. Generators Sorted on Descending Order of Popularity Per Section 2.2

ID	Name	License	Status Date	Platform[5]	Category	Link (Source, Binaries, Paper)
1	iPerf2 [99]	BSD	2019-01	All	CMT[2]	https://sourceforge.net/projects/iperf2/
2	Netperf [78]	Free[1]	2018-06	All	CMT[2]	https://hewlettpackard.github.io/netperf/
3	httperf [102]	GPLv2	2018-11	All	App level gen	https://github.com/httperf/httperf
4	moongen [47]	MIT	2018-12	Unix, Linux	Script driven	https://github.com/emmericp/MoonGen
5	scapy [33]	GPLv2	2019-01	All	Script driven	http://www.secdev.org/projects/scapy/
6	Linux pktgen [109]	GPLv1	2018-09	linux	Script driven	https://github.com/torvalds/linux/blob/master/net/core/pktgen.c
7	netcat [76]		2019-01	All	Other[3]	http://nc110.sourceforge.net/
8	iperf3 [91]	BSD-3-Clause	2018-12	All	CMT[2]	https://github.com/esnet/iperf
9	TCPreplay [5]	GPLv3	2018-12	All	Traffic replay	http://tcpreplay.appneta.com/
10	DPDK pktgen [149]	BSD	2019-01	Unix, Linux	Script driven	https://pktgen-dpdk.readthedocs.io/en/latest/
11	Harpoon [132]	GPLv2	2018-01	Unix, Linux	Trace driven	https://github.com/jsommers/harpoon
12	D-ITG [27]	GPLv3	2013-03	All	Model based	http://www.grid.unina.it/software/ITG/
13	TMIX [148]	MIT	2011-11	NS2 or NS3	Other[3]	https://github.com/weiglemc/tmix-ns2
14	Nuttcp [51]	GPLv2	2018-07	All	CMT	https://www.nuttcp.net/
15	SWING [143]	Free[1]	2008-09	Unix, Linux	Trace driven	http://cseweb.ucsd.edu/~kvishwanath/Swing/
16	Surge [30]	Free[1]	1998-11	All	App level gen	http://cs-www.bu.edu/faculty/crovella/surge_1.00a.tar.gz
17	OSNT [4]	-	2019-01	NetFPGA	Script driven	http://osnt.org/
18	Bit-Twist [152]	GPLv2	2012-04	All	Traffic replay	http://bittwist.sourceforge.net/
19	Globetraff [82]	-	2016-09	All	Trace driven	https://github.com/lookat119/GlobeTraff
20	Ixnetwork [85]	Commercial	-	-	-	https://www.ixiacom.com/products/ixnetwork
21	Nping [107]	GPLv2	2018-03	All	CMT	https://nmap.org/nping/
22	TRex [37]	Apache-v2	2019-01	Unix, Linux	Script driven	https://trex-tgn.cisco.com/
23	Ostinato [113]	GPLv3	2019-01	All	Script driven	https://ostinato.org/
24	libcrafter [118]	MIT	2017-09	Unix, Linux	Script driven	https://github.com/pellegre/libcrafter
25	PackMime-HTTP [147]	MIT	2005-06	NS2	App level gen	https://www.cs.odu.edu/~mweigle/research/packmime/
26	Spirent SmartBits [134][6]	Commercial	-	-	-	https://www.spirent.com/products/testcenter

(Continued)

Table 1. Continued

ID	Name	License	Status Date	Platform[5]	Category	Link (Source, Binaries, Paper)
27	Nemesis [104]	GPLv2	2003-11	All	Script driven	http://nemesis.sourceforge.net/
28	LANforge FIRE [35][6]	Comm-ercial	-	All	-	http://www.candelatech.com/
29	Mtools [26]	-	-	-	-	http://www.grid.unina.it/grid/mtools/
30	Netspec [79]	-	1997-12	Unix, Linux	Trace driven	http://www.ittc.ku.edu/netspec/
31	Skaion TGS [130]	Comm-ercial	-	-	Other	http://www.skaion.com/
32	Trafgen [75]	GPLv2	2019-01	Unix, Linux, Mac	App level gen	http://netsniff-ng.org/
33	RAMP [94]	-	-	-	Trace driven	http://www.csie.ncku.edu.tw/~klan/data/materials/ramp.pdf
34	BRUTE [34]	GPLv2	2016-11	Linux	CMT[2]	https://github.com/awgn/brute
35	Breaking-Point [83]	Comm-ercial	-	-	App level gen	https://www.ixiacom.com/products/breakingpoint-ve
36	IP-Packet [29]	GPLv2	2003-11	Linux, FreeBSD	CMT[2]	http://p-a-t-h.sourceforge.net/html/index.php
37	Rude/ Crude [93]	GPLv2	2002-06	All	CMT[2]	http://www.atm.tut.fi/rude
38	Bruno [3]	-	-	-	Trace driven	https://ieeexplore.ieee.org/document/4667607
39	Divide & Conquer [151]	-	-	-	Traffic replay	https://doi.org/10.1109/TRIDNT.2005.18
40	Byte-Blower [49]	Comm-ercial	-	-	-	https://www.excentis.com/products/byteblower/
41	Colosoft Packet Builder [77]	Free[1]	2016-06	Windows	CMT[2]	http://www.colasoft.com/download/products/download_packet_builder.php
42	EAR Replay [89]	-	-	-	Traffic replay	https://doi.org/10.1109/WCNC.2012.6214199
43	GL traffic generator [74]	Comm-ercial	-	-	-	https://www.gl.com/traffic-generators.html
44	HexInject [1]	BSD-2-Clause	2017-01	Linux	CMT[2]	http://hexinject.sourceforge.net/
45	IPGen [97]	-	2001-03	-	CMT[2]	http://sourceforge.net/projects/ipgen/
46	IxChariot [84]	Comm-ercial	-	-	Trace driven	https://www.ixiacom.com/products/ixchariot
47	PIM-SM Packet Generator [2]	-	-	-	-	https://literature.cdn.keysight.com/litweb/pdf/5988-6560EN.pdf?id=1649878
48	EPB [141]	Free[1]	2019-05	All (C)	Script driven	http://m-a-z.github.io/epb/
49	NETI@ home [128]	-	-	-	-	http://neti.gatech.edu/
50	TTCP, Test TCP [38]	-	-	-	-	https://www.cisco.com/c/en/us/support/docs/dial-access/asynchronous-connections/10340-ttcp.html
51	LANTraffic [40]	Comm-ercial	2015-11	Windows	CMT[2]	https://www.zti-communications.com/lantrafficv2/

(Continued)

Table 1. Continued

ID	Name	License	Status Date	Platform[5]	Category	Link (Source, Binaries, Paper)
52	Libtins [52]	BSD	2019-01	All	Script driven	https://github.com/mfontanini/libtins
53	LitGen [123]	-	-	-	Trace driven	https://dl.acm.org/doi/10.5555/1762888.1762896
54	MGEN [92]	MIT-ish	2018-11	All	Model based.	https://www.nrl.navy.mil/itd/ncs/products/mgen
55	UDP Generator [136]	MIT	1999-05	Linux, Unix	CMT[2]	http://www.citi.umich.edu/projects/qbone/generator.html
56	Network Expect [46]	-	-	Unix, Linux, Mac	CMT[2]	http://www.netexpect.org/
57	Cat Karat [43]	Commercial	2010-01	Windows	CMT[2]	https://sites.google.com/site/catkaratpacketbuilder/
58	NTG [153]	-	-	-	App level gen	http://www.wseas.us/e-library/conferences/2013/Paris/CCTC/CCTC-35.pdf
59	Fragout [133]	BSD-3-Clause	2002-04	All	CMT[2]	http://www.monkey.org/~dugsong/fragroute/
60	GEIST [80]	BSD-2-Clause	2012-11	All	Model based	http://kkant.net/geist/
61	NTGM [117]	Commercial	2018-10	Windows	CMT[2]	http://pbsftwr.tripod.com/id17.html
62	Graph-Based TG [129][4]	-	-	-	-	http://rvs.unibe.ch/research/pub_files/SSKB10.pdf
63	Internetworking Test TG [44][4]	-	-	-	-	http://www.donfraysoftware.com/MITS/MITS.htm
64	Omnicor TG [110][4]	Commercial	-	-	Model based	https://www.omnicor.com/products/network-testing-tools
65	Jugi's TG [98][4]	GPLv2	2010-11	Linux	CMT[2]	http://www.netlab.tkk.fi/~jmanner/jtg.html
66	KUTE [154]	GPLv2	2007-09	Linux	CMT[2]	http://caia.swin.edu.au/genius/tools/kute/
67	LAN-decoder32T [139]	-	-	-	-	http://www.triticom.com/triticom/ld32/trafgen.htm
68	packet sender [103]	GPLv2	2018-12	All	CMT[2]	https://packetsender.com/
69	PackETH [116]	GPLv3	2017-12	All	CMT[2]	http://packeth.sourceforge.net/
70	Mausezhan [146]	GPLv2	2011-12	Linux (C)	CMT[2]	https://github.com/uweber/mausezahn
71	MxTraf [88]	GPLv2	-	-	-	http://mxtraf.sourceforge.net/
72	Solarwinds WAN killer [131]	Commercial	-	Windows	CMT[2]	https://www.solarwinds.com/topics/traffic-generator-wan-killer
73	NSWEB [145]	-	-	NS2	-	https://www.net.t-labs.tu-berlin.de/~joerg/
74	NTGen [28]	-	2002-11	Linux (C/C++)	-	http://softlab-pro-web.technion.ac.il/projects/NTGen/html/ntgen.htm
75	STG-10G [45]	Commercial	-	-	Model based	https://www.ecdata.com/products/stateful-traffic-generator/

(Continued)

Table 1. Continued

ID	Name	License	Status Date	Platform[5]	Category	Link (Source, Binaries, Paper)
76	PacGen [114]	GPL-v2	2006-09	Linux (C)	CMT[2]	http://sourceforge.net/projects/pacgen/
77	PlayCap [112]	GPLv3	2010-03	All	Traffic replay	https://github.com/signal11/PlayCap
78	Poisson TG [122][4]	-	2003-06	(C)	Model based	http://www.spin.rice.edu/Software/poisson_gen/
79	ProvaGEN 3.0 [42]	-	-	-	-	http://www.provanet.com/packet_generator_tts_page.htm
80	Qosnetics TG [126][4]	-	-	-	-	http://www.qosnetics.com/
81	Real-Time Voice TG [125][4]	-	-	-	-	http://www.cs.ucr.edu/~msamidi/projects.htm
82	VOIP TG [32][4]	-	2005-11	(perl)	App level gen	http://voiptg.sourceforge.net/
83	Self Similar TG [87][4]	MIT	2001-04	(C)	Model based	http://research.glenkramer.com/code/trf_gen3.shtml
84	Sources-OnOff [140]	GPLv3, CeCILL	2013-03	Linux (C)	Model based	http://www.recherche.enac.fr/~avaret/sourcesonoff
85	SPAK [81] Packet Generator	-	-	-	-	http://static.lwn.net/lwn/1998/0312/a/spak.html
86	TCPivo [50]	-	2002-09	Linux (C)	Traffic replay	https://www.thefengs.com/wuchang/work/tcpivo/
87	TfGen [137]	-	1998-02	Windows	CMT[2]	http://www.pgcgi.com/hptools/
88	IP-traffic [115]	Comm-ercial	2019	Windows	CMT[2]	https://www.pds-test.co.uk/products/ip_test_measure.html
89	Traffic Generator Tool [119]	-	-	-	-	http://www.postel.org/tg/
90	WRAP [106]	BSD clause	2019-01	All	Script driven	https://github.com/Juniper/warp17
91	Yersinia [31]	GPLv2	2017-09	All	CMT[2]	https://github.com/tomac/yersinia
92	YouTube Workload generator [36]	-	-	-	-	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.471.4292&rep=rep1&type=pdf

[1]Some traffic generators classified as free require attribution.
[2]CMT stands for constant or maximum throughput traffic generators (see Section 3).
[3]Other, when not listed in the pre-defined traffic generator categories in Section 3.
[4]TG is used as an abbreviation for traffic generator.
[5]Platform refers to supported operating systems.
[6]Hardware traffic generators, all others are software traffic generators.

constant/max throughput traffic generators, even when controlling for the smaller set of recent publications. Although they do not make the top 10, there are many of other realistic traffic generators in the next 10 on the list in Table 1.

In recent years usage of script driven traffic generators like `moongen` [47], and `DPDK pktgen` [149] that allow extensive variations in specific header values have gained more traction, and we expect that trend to continue. In addition, some of these script driven traffic generators like `trex` [37] do not feature among the top 10. However, we believe that generators of this type will find increasing utility in research in the near term.

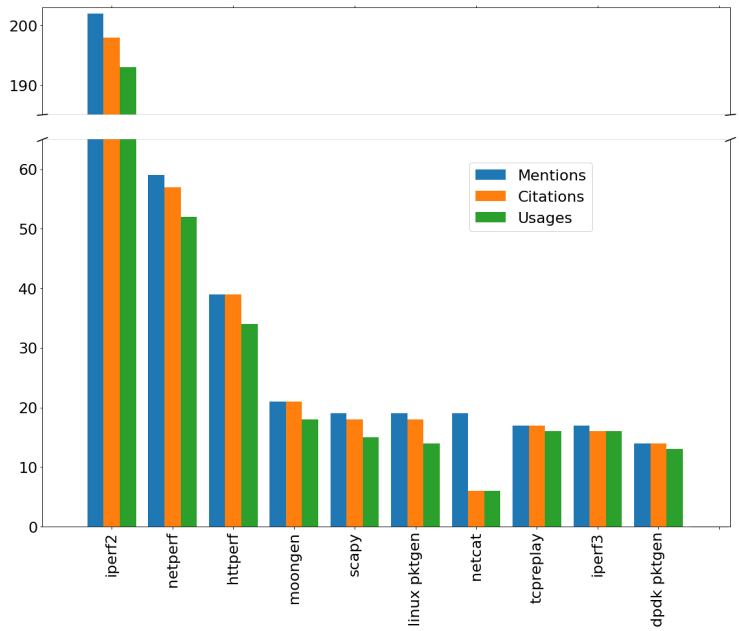

Fig. 1. Top traffic generators from ACM and USENIX networking-related conference publications [121].

Most constant/maximum throughput traffic generators create packets with almost no variation in header and payload contents—they usually only allow for selection of a single value for specific header fields before the beginning of their generation process. As such, they are useful for a narrow class of applications and may not reflect a true mix of network traffic in typical topologies. For example, one such generator is `iperf2`. Per each run, `iperf2` can provide a TCP or UDP flow that is driven by a constant throughput goal. Despite these limitations, these types of generators are quite popular. We examined all papers where `iperf2` was used to find out if such papers used any other traffic generation mechanisms in combination, but found almost none. The count of the number of papers per year in which `iperf2` was used exclusively is plotted in Figure 3, mirroring the baseline numbers in Figure 2.

We need to note that bespoke generation techniques are not included in our analysis. In some types of experiments that require specific traffic patterns, researchers may write appropriate wrapper scripts and native packet creators for the generation task with the desired traffic patterns. Nevertheless, there were not a significant number of papers referring to traffic generation without a reference to a specific generator.

3 A TAXONOMY OF TRAFFIC GENERATORS

Traffic generators are software tools or hardware devices that create network packets based on scripted, recorded, or configured patterns. That is, packets from traffic generators are not generated through actual application conversation workflows, even though the traffic generation system may be designed to generate packets that mimic what may be seen in a real production environment. Different network traffic generators are designed with different goals. While some are designed to stress-test network devices and software, some others are designed to be able to craft packets for tests of performance and behavior correctness. In this survey we focus on the behavior analysis of traffic generators, as performance is strongly influenced by individual deployment environments.

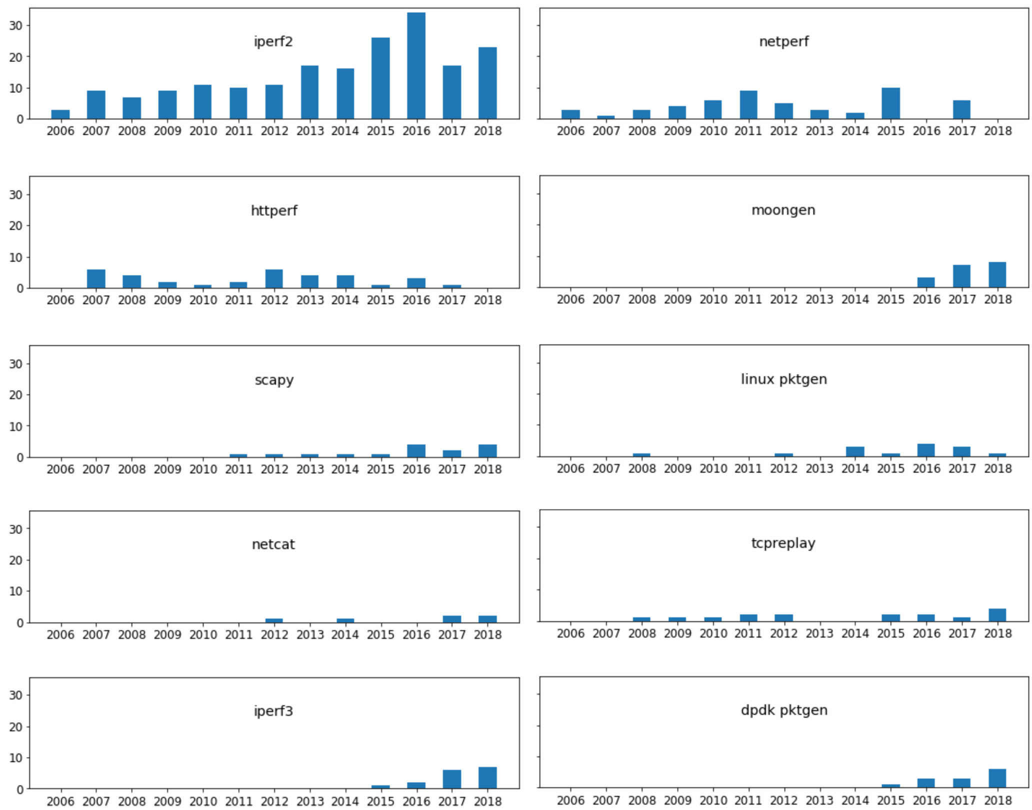

Fig. 2. Usage by year of top 10 generators as cited in ACM and USENIX networking-related conference publications.

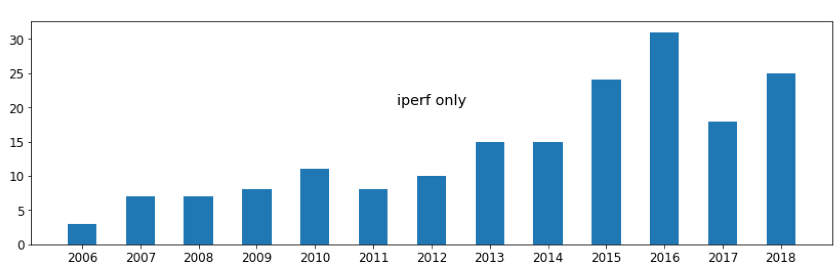

Fig. 3. Per year exclusive usage citations of `iperf2`.

We expand on the classifications provided by Molnár et al. [101] from the perspective of their techniques for pushing packets into the network. Traffic generators can be categorized into constant/maximum throughput generators, application-level synthetic workload generators, trace file replay systems, model-based generators, and script driven traffic generators. We describe each class below.

3.1 Constant or Maximum Throughput Generators

A packet is created with specific application-layer header fields and then repeatedly sent out on a network interface at a constant rate or at the maximum possible rate (in **bits per second** (**bps**) or

Table 2. Classification of Traffic Generators per Section 3

	Traffic Generator Category	All
1	Constant or maximum throughput generators	26
2	Application level generators	7
3	Trace file replay tools	7
4	Model-based traffic generators	11
5	Trace driven model-based traffic generators	5
6	Script driven traffic generators	11
7	Others	25
	Total	92

packets per second). Popular examples of traffic generators in this category are `iperf2` [138] and `netperf` [78]. These are often the easiest to use, and are suitable for quick network throughput stress testing. A characteristic of generators in this category is that they offer little or no variation in header and payload content of the packets that are blasted out the interfaces. In most traffic generators in this class—while some are capable of preserving the semantics of connection-oriented network layer (TCP, etc.)—a user can specify only a single set of flow parameters per running instance (such as the source and destination IP address and port numbers).

3.2 Application Level Synthetic Workload Generators

These generate network packet traffic for a specific type of application or higher-layer protocol such as the `httperf` [102]. In some cases, researchers may launch actual application programs and run a specific set of workloads using their data exchanges to generate the traffic. This approach of workload generation is often capable of realistic variations on packets for the specific application or protocol. However, the resulting workload still consists of a limited set of application events. Exclusive usage of these approaches may result in network behavior deprived of realistic simultaneous background traffic or the application-user interaction in a typical production environment [142].

3.3 Trace File Replay Systems

Replay systems inject packets from a pre-existing trace file into a network interface at the indicated time intervals in the capture file. In some cases, users are able to specify the speed at which they would like to replay the packets. Many researchers obtain trace files with anonymous data and empty payload contents from public data sets [53, 95, 124], and replay them on the nodes of their individual experiment topologies using tools like `TCPreplay` [5]. These replay systems can produce traffic workloads that mirror the original traffic, especially if the workload can be run on an experiment topology that is similar to the original network. However, most replay systems are stateless and are unable to send the packets in a manner that will be responsive to network congestion or topology events in the experiment. For example, such a replay will continue to send out TCP packets even when the links between endpoints are down whereas a realistic TCP flow control would have limited further packet transmissions. In addition, continuous replay of the same trace file on a network will keep producing the same events periodically resulting in traffic patterns that are unrealistic for many use cases.

3.4 Model-Based Traffic Generators

Given that network traffic is quite random, bursty, and self similar [73, 96, 150], a popular method of generating realistic traffic is the creation and transmission of packets following random distributions of their time intervals, packet sizes, and so on. One example of these is the **Multi-Generator** (**MGEN**) [92] traffic generator. These generators allow users to specify a random distribution model with parameters that may match the intended scenario of network traffic workload. With carefully selected random distributions, they can generate traffic that is statistically similar to traffic workloads in specific production environments.

3.5 Trace Driven Model-Based Traffic Generators

Some traffic generators go a step further than the purely model-based approach by allowing experimenters to supply a trace file input or log files of actual traffic from production networks. The input trace file or log file is analyzed to create a model by fitting the various traffic parameters to random distributions which are then used to generate packets. Good examples are `harpoon` [132] and `swing` [144]. They generate packets that are statistically similar to actual packets seen in the corresponding input production network trace or log file.

3.6 Script Driven Traffic Generators

In recent years many new scripts driven traffic generators have been developed. These generators allow users to dynamically modify the full range of packet header and data content through complex coded logic. Popular examples of generators in this category are `DPDK pktgen` [149] and `moongen` [47]. These allow users to create any type of packet, with almost any packet header value, and while also dynamically modifying the packets at run time.

4 TRAFFIC GENERATOR SELECTION: COMMON REQUIREMENTS AND FEATURES

Traffic generators have diverse sets of features and they typically report a small set of built-in metrics. There is no single traffic generator that is better in all experiment use cases than every other one in terms of serving a research objective through these features and reported metrics. For instance, while a particular traffic generator may be good at injecting packets into a network at very high speeds, it may not provide dynamic packet length variations. In this respect, our analysis of capabilities of generators resulted in a structured digests of features as presented below in tabular form.

In Table 3, we show whether there is support for a particular feature among the top 10 traffic generators in our survey. We examined the experiments, evaluations and methodology sections of the surveyed papers for the research goals, the types of traffic workloads and their corresponding required features. We then examined the documentation, source code, man pages, help information, and the associated research papers for each of the generators in the top 10 list to verify the presence of a particular capability in the Table 3.

Table 4 further gives a list of the header fields in the Ethernet, IPv4, TCP, and UDP protocol stacks, and provides information on how each of the traffic generators in the top 10 list supports the configuration of that header field. In some cases, header fields can be set only to a constant value, while in other cases header fields can be set to vary within a range or be fully randomized during the packet generation process.

Table 5 presents a list of common metrics among the auto-generated reports of traffic generators. No single traffic generator reports a set of metrics that would be regarded as comprehensive, and there are some that do not give a report at all. It is important to note that even when a generator is able to report metrics, the limitations of execution environments may affect timing information or strip low-level header data before reaching the generator, introducing uncontrolled error. Therefore, it is always advised to validate metrics through packet traces of the generated traffic that are captured on the wire.

Table 3. Traffic Generator Selection: Common Features and Experiment Requirements

		Generator									
	Feature[1]	iperf2	net-perf	http-erf	moon-gen[2]	scapy	Linux pkt-gen[2]	netcat	iperf3	TCP-replay	DPDK pkt-gen[2]
1	set # of packets					✓	✓		✓	✓	✓
2	set total bytes	✓				✓			✓		
3	set fixed throughput	✓[3]			✓	✓	✓		✓	✓	✓
4	set randomized throughput				✓	✓					
5	set packet rate				✓	✓	✓			✓	✓
6	set time duration	✓	✓		✓	✓	✓		✓	✓	
7	send data files		✓			✓		✓	✓		
8	replay traffic traces				✓	✓				✓	
9	set fixed packet size				✓	✓		✓			✓
10	set randomized packet sizes				✓	✓	✓				✓
11	set fixed inter-packet time				✓	✓					
12	set randomized inter-packet times				✓	✓					
13	support TCP connections	✓	✓	✓		✓		✓	✓		
14	support SCTP connections		✓			✓			✓		
15	set MSS	✓				✓			✓		
16	set reporting intervals	✓	✓		✓	✓			✓	✓	✓
17	set interface				✓	✓	✓			✓	✓
18	specify IP addr. of interface	✓	✓					✓	✓		
19	set CPU affinity		✓		✓		✓		✓		✓
20	generate IP fragments				✓	✓	✓				✓
21	bi-directional generation	✓	✓	✓		✓					
22	multiple parallel connections/flows	✓	✓	✓	✓	✓	✓		✓		✓
23	arbitrary http requests			✓		✓		✓			

[1]Feature descriptions are provided in appendix A.1.
[2]Requires exclusive control of the network interface.
[3]UDP only.

4.1 Traffic Generator Selection Methodology

The Table 3 presents the top 10 list with varying levels of support for the list of features. There are some generators that support all of the features in addition to giving a comprehensive set of metric reports, but they usually require commercial licenses that are quite expensive [85, 130]. Researchers are tasked with a preliminary assessment of generator features and an evaluation of each generator for the research objectives. We provide a method to select a traffic generator using this paper's compilations on generator categories and their respective features. As features of each generator advances and as new generators are created, we expect that there will be the community updates on the tabular digests.

(1) **Definition of Workload Requirements:** Before selecting a generator, an experimenter first needs to identify specific requirements of the traffic workload that are of vital importance for the experimental goals of the research. Based on the traffic generation objectives,

Table 4. Supported Configuration of Header Fields for the Top 10 Traffic Generators

		Generator									
	Header Field	iperf2	net-perf	http-erf	moon-gen	scapy	Linux pktgen	netcat	iperf3	TCP-replay	DPDK pktgen
1	L2 source MAC				★	★	★			★	★
2	L2 destination MAC				★	★	★			★	★
3	L2 VLAN ID				★	★	✓			★	★
4	L2 ethertype				★	★				★	★
5	L3 source IP	✓	✓		★	★	★	✓	✓	★	★
6	L3 destination IP	✓	✓	✓	★	★	★	✓	✓	★	★
7	L3 header length				★	★				★	★
8	L3 DSCP/TOS	✓			★	★	✓	✓	✓	★	★
9	L3 ECN				★	★				★	★
10	L3 total length				★	★				★	★
11	L3 identification				★	★				★	★
12	L3 don't fragment				★	★				★	★
13	L3 more fragments				★	★				★	★
14	L3 fragment offset				★	★				★	★
15	L3 TTL	✓			★	★				★	★
16	L3 protocol				★	★				★	★
17	L3 header checksum				★	★				★	★
18	L4 source port		✓		★	★	★[1]	✓	✓	★	★
19	L4 destination port	✓	✓	✓	★	★	★[1]	✓	✓	★	★
20	TCP sequence num				★	★				★	★
21	TCP ack number				★	★				★	★
22	TCP data offset				★	★				★	★
23	TCP reserved bits				★	★				★	★
24	TCP flags				★	★				★	★
25	TCP window size	✓	✓		★	★			✓	★	★
26	TCP checksum				★	★				★	★
27	TCP urgent pointer				★	★				★	★
28	TCP options	✓[2]	✓[2]		★	★			✓[2]	★	★
29	UDP length				★	★				★	★
30	UDP checksum				★	★				★	★

✓: set to single value (no variation of the header field is supported during generation)
★: single, varying, or randomized values can be set for the header field
[1]UDP only.
[2]TCP_NODELAY option only.

the researcher must then identify the specific features of the desired traffic workload picked from the Tables 3 and 4. This first step is in a sense the most important one in the methodology. The chosen requirements will serve as the driving input for each of the subsequent steps below.

(2) **Availability:** A researcher may start with the list of 92 traffic generators given in the Table 1 to determine which generators are available. We note that the generator has to be within the skill-set, resources, and capabilities of the researcher from the perspectives of the research platform requirements and ease-of-use.

(3) **Validation of the Workload:** The initial list is then filtered based on the category of traffic workloads. A taxonomy for the workload characteristics has been provided in Section 3. The specific requirements of the traffic workload can then be validated by the characteristics of the generators of choice.

(4) **Features:** One of the key decision points for the researcher is what features are supported by a generator of choice (Tables 3 and 4). The desired workload properties for the specific

Table 5. Reported Metrics for the Top 10 Generators

		Generator									
	Reported Metric[1]	iperf2	net-perf	http-erf	moon-gen	scapy	Linux pktgen	netcat	iperf3	TCP-replay	DPDK pktgen
1	throughput	✓	✓	✓	✓		✓		✓	✓	✓
2	latency		✓		✓		✓				✓
3	packet rate				✓		✓			✓	✓
4	total no. of packets	✓			✓		✓		✓	✓	✓
5	total no. of bytes	✓	✓		✓				✓	✓	✓
6	duration	✓	✓	✓	✓		✓		✓	✓	✓
7	jitter	✓			✓				✓		✓
8	no. of retransmissions		✓		✓					✓	
9	no. of drops	✓			✓		✓		✓	✓	✓
10	MSS	✓	✓		✓						
11	congestion win. size(s)	✓							✓		
12	CPU demand		✓	✓							
13	number of flows or connections	✓	✓	✓	✓		✓		✓	✓	✓
14	request/response transaction rates		✓	✓[2]							

[1]Metric descriptions are provided in appendix A.2.
[2]http only.

research goals could result in a trade-off when generators that lack some of the key features are preferred per availability, ease-of-use, performance and other concerns.

The tabulated digests provide a guideline in picking features to address traffic generation objectives of the researcher. We expect that these digests will be kept up to date with the new developments on existing generators and as new generators are added while preserving most of the structure.

4.2 An Example: Load-Balancing Research

To demonstrate a typical walk-through of the steps in the previous section, we present a research project on a new fictitious layer 4 load-balancing network function that leverages TCP header information.

Step 1 - Requirements: The traffic workload required for this research has diverse transaction characteristics with relatively fixed total throughput. Therefore, the features are:

(1) multiple parallel TCP connections or flows;
(2) ability to generate packets with varying packet sizes;
(3) ability to vary header fields: L3 source IP addresses and L4 source port numbers.

Step 2 - Availability: Based on the constraints of our runtime environment, licensing requirements, and current availability of each generator, we filter the options in Table 1 down to 31 choices. Among this number we have all those in the top 10 list—`iperf2`, `netperf`, `httperf`, `moongen`, `scapy`, `linux pktgen`, `netcat`, `TCPreplay`, `iperf3`, `DPDK pktgen`, and 21 others.

Step 3 - Characteristics of Desired Traffic: Many classes of traffic generators have support for diverse transactions and relatively fixed total throughput. For example, some constant/maximum throughput traffic generators (Section 3.1) allow for multiple simultaneous TCP and UDP connections. Many model-based generators (Section 3.4 and 3.5) and script driven traffic generators (Section 3.6) also support the type of traffic desired.

Step 4 - Features: Tables 3 and 4 are utilized to narrow down the generators of interest for the research task. Based on the feature list in step-1, shortlisted generators must have check marks on rows 3, 10, and 22 of Table 3 and a star in rows 5 and 18 of Table 4. Thus, the resulting list includes `scapy`, `moongen`, and `dpdk pktgen`. We do not include `linux pktgen` in the shortlist because randomization of L4 source port numbers is not possible as shown in footnote 1 of Table 4, even though all other requirements are met by this generator.

Further discrimination among the selected tools will require experiment-specific considerations.

5 EXISTING SURVEYS

There have been a few surveys on network traffic generators in the past. Kolahi et al. [86] evaluated the TCP throughput performance of four traffic generators, and provided a comparison of their features. The generators compared include Iperf, Netperf, D-ITG, and IP Traffic. The experiments were exclusively carried out between 2 computers running the Windows operating system. Authors observed that the bandwidth that the tools measure can vary as much as 16.5 Mbps for a TCP connection over a 100 Mbps link. For the same network set up, Iperf measured the highest bandwidth (93.1 Mbps) while IP traffic the lowest (76.7 Mbps).

Molnár et al. [101] unveiled the fact that there is no consensus in the research community how to validate network traffic generators. They recommended 9 metrics that could be used to validate traffic generators and classified 19 traffic generators into 5 classes, presenting specific validation techniques for each class of generators.

Mishra et al. [100] compared six traffic generators in terms of TCP and UDP throughput performance under various scenarios. The generators compared include D-ITG, PackETH, Ostinato, Iperf, Netperf, and IP Traffic. Their results showed different generators excelled in terms of various metrics under different circumstances. They concluded that a single traffic generator is not applicable for all types of networks. Traffic Generators are designed for specific applications depending upon the need and characteristics of application and network.

More recently, Emmerich et al. [48] undertook a performance comparison for high-performance software traffic generators in terms of their approach to rate control, performance, precision, and accuracy of packet injection times. The traffic generators compared include Moongen, DPDK Pktgen, Linux Pktgen, and pfq-gen. They observed that for most of these software based high performance generators can offer good thoughput performance and high accuracy of packet injection times as long as overloading does not occur. The work also showed that the performance and precision of most high frequency software generators is greatly dependent on clock frequency of the CPU hardware used.

Most of the surveys cited above directly compare the performance of a selected short list of traffic generators. We realize that each generator may have unique features that make it more suitable for specific types of traffic generation than the others. Therefore, we do not attempt to directly compare the performance of the traffic generators—something that should be validated independently in each new experiment configuration—but we present their features along with a methodology to qualitatively make a shortlist of traffic generators that meet the requirements for specific experimental objectives.

6 SUMMARY

We present a survey that identifies 92 traffic generators from a large corpus of conference proceeding publications. We perform a classification of the generators based on the method of traffic generation. From the results of our survey, we determine the top 10 most popular traffic generators through analysis of over 7,000 papers published in ACM, SIGCOMM, and USENIX conferences over the last 13 years. We observe that the set of supported features by each traffic generator vary considerably. By determining the main functionality of each generator at hand, we categorize features for individual generators into a structured form to eventually serve research objectives. Our compilations provide the traffic generator outcomes and functionality, which are then used in the traffic generation selection mechanism in Section 4.1. The compilations in the tables are expected to be updated as individual generator functionalities evolve and new generators are released. However, the methodology will stay the same for the alignment of experimental objectives to the choice of the generator capabilities.

APPENDIX

A DEFINITIONS OF TABLE ROW HEADERS

A.1 Table 3

The descriptions for each feature listed in Table 3 are given below.

1 **Set # of packets:** Configure the total number of packets to send.
2 **Set total bytes:** Configure the total number of bytes to send.
3 **Set fixed throughput:** Set a fixed value for the throughput in bps
4 **Set randomized throughput:** Configure set of values, or a random distribution for the throughput at which to send packets.
5 **Set packet rate:** Configure a fixed value in **packet rates per second** (**pps**) at which packets should be sent.
6 **Set time duration:** Set a time limit for the duration of the traffic generation process.
7 **Send data files:** Configure the generator to use an arbitrary data file as data source for the payload of the packets to be sent.
8 **Replay traffic traces:** Generator supports the replay network traffic trace files.
9 **Set fixed packet size:** Configure a packet size in bytes, for all packets to be sent by the generator.
10 **Set randomized packet sizes:** Configure packet sizes to be picked from a set of values. These values can be picked from a particular random distribution.
11 **Set fixed inter-packet time:** Set a fixed value for inter-packet time intervals in seconds for the packets.
12 **Set randomized inter-packet times:** Configure inter-packet time values to be picket from a set of values or from a random distribution.
13 **Support TCP connections:** Generator supports actual TCP connections, and not just 1-sided flows.
14 **Support SCTP connections:** Generator supports actual SCTP connections, and not just 1-sided flows.
15 **Set MSS:** Configure a fixed value for **maximum segment size** (**MSS**).
16 **Set reporting intervals:** Configure time intervals at which to show a summary of the packets sent so far, while the generation process is ongoing.
17 **Set interface:** Select the network interface on which to send out packets.
18 **Specify IP address of interface:** Select the interface on which to send out packets, by specifying the IP address associated with the interface.
19 **Set CPU affinity:** Select a CPU core to use for the packet generation process on multi-core systems.
20 **Generate IP fragments:** Native support for the generation of fragmented IP packets.
21 **Bi-directional generation:** Native support for sending packets in both directions, from the source and the target, each one towards the other.

22 **Multiple parallel connections/ flows:** Native support for sending packets associated with multiple flows or connections simultaneously.
23 **Arbitrary http requests:** Configure to send any HTTP request to a target host.

A.2 Table 5

The descriptions for each feature listed in Table 5 are given below.

1 **Throughput:** The amount of data delivered by the traffic generator from the source to target per unit time, usually measured in bps.
2 **Latency:** The interval between the time a packet is sent from a source, and the time it is received at the destination.
3 **Packet rate:** The number of packets delivered by the traffic generator from the source to target, usually measured in pps.
4 **Total no. of packets:** The total number of packets sent from the source to the target during the entire traffic generation process.
5 **Total no. of bytes:** The total amount of data in bytes sent from the source to the sink during the traffic generation process.
6 **Duration:** The total time elapsed during the traffic generation process usually measured in seconds.
7 **Jitter:** The variation in latency of packets usually measured in seconds.
8 **No. of retransmissions:** The total number of packets that had to be re-transmitted during the packet generation process.
9 **No. of drops:** The total number of packets that were sent from the source but not successfully received at the receiver.
10 **MSS:** The MSS of TCP packets sent by the generator.
11 **Congestion win. size(s):** The congestion window size of the sending host of the traffic generator.
12 **CPU demand:** The amount of CPU utilized by the traffic generator.
13 **Number of flows or connections:** The total number of unique connections or the total number of unique flows created by the traffic generation process.
14 **Request/response transaction rates:** For the traffic generators that conform to the request-response model, this is the number of request and response pairs completed per unit time.

REFERENCES

[1] Emanuele Acri. 2017. HexInject: The Power of Raw Hex Network Access. Retrieved 13 April, 2020 from http://hexinject.sourceforge.net/.
[2] Agilent Technologies. 2002. PIM-SM Multicast Performance Testing. Retrieved 12 April, 2020 from https://literature.cdn.keysight.com/litweb/pdf/5988-6560EN.pdf?id=1649878.
[3] Gianni Antichi, Andrea Di Pietro, Domenico Ficara, Stefano Giordano, Gregorio Procissi, and Fabio Vitucci. 2008. Bruno: A high performance traffic generator for network processor. In *Proceedings of the Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on.* IEEE, 526–533.
[4] Gianni Antichi, Muhammad Shahbaz, Yilong Geng, Noa Zilberman, Adam Covington, Marc Bruyere, Nick Mckeown, Nick Feamster, Bob Felderman, Michaela Blott, Andrew Moore, and Philippe Owezarski. 2014. OSNT: Open Source Network Tester. *IEEE Network* 28, 5 (Sept. 2014), 6–12. DOI: https://doi.org/10.1109/MNET.2014.6915433
[5] AppNeta. 2011. *Tcpreplay.* AppNeta Inc. Retrieved from https://tcpreplay.appneta.com/.
[6] USENIX (Advanced Computing Systems Association). 2006-2007. In *Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet.* (all published papers in the proceedings).
[7] USENIX (Advanced Computing Systems Association). 2006–2012. In *Proceedings of the USENIX Summit on Hot Topics in Security.* (all published papers in the proceedings).
[8] USENIX (Advanced Computing Systems Association). 2006–2013. In *Proceedings of the Large Installation System Administration Conference.* (all published papers in the proceedings).

[9] USENIX (Advanced Computing Systems Association). 2006–2014. In *Proceedings of the Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. (all published papers in the proceedings).

[10] USENIX (Advanced Computing Systems Association). 2006–2018. In *Proceedings of the USENIX Annual Technical Conference*. (all published papers in the proceedings).

[11] USENIX (Advanced Computing Systems Association). 2006–2018. In *Proceedings of the USENIX Security Symposium*. (all published papers in the proceedings).

[12] USENIX (Advanced Computing Systems Association). 2006–2018. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*. (all published papers in the proceedings).

[13] USENIX (Advanced Computing Systems Association). 2006–2018. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. (all published papers in the proceedings).

[14] USENIX (Advanced Computing Systems Association). 2007. In *Proceedings of the International Workshop on Networking Meets Databases*. (all published papers in the proceedings).

[15] USENIX (Advanced Computing Systems Association). 2008. In *Proceedings of the USENIX Workshop on the Analysis of System Logs*. (all published papers in the proceedings).

[16] USENIX (Advanced Computing Systems Association). 2008. In *Proceedings of the Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. (all published papers in the proceedings).

[17] USENIX (Advanced Computing Systems Association). 2008–2018. In *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test*. (all published papers in the proceedings).

[18] USENIX (Advanced Computing Systems Association). 2009–2010. In *Proceedings of the International Workshop on Peer-to-Peer Systems*. (all published papers in the proceedings).

[19] USENIX (Advanced Computing Systems Association). 2009–2018. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing*. (all published papers in the proceedings).

[20] USENIX (Advanced Computing Systems Association). 2010. In *Proceedings of the Asia-Pacific Workshop on Systems*. (all published papers in the proceedings).

[21] USENIX (Advanced Computing Systems Association). 2010–2012. In *Proceedings of the USENIX Conference on Web Application Development*. (all published papers in the proceedings).

[22] USENIX (Advanced Computing Systems Association). 2011–2018. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. (all published papers in the proceedings).

[23] USENIX (Advanced Computing Systems Association). 2014. In *Proceedings of the Open Networking Summit*. (all published papers in the proceedings).

[24] USENIX (Advanced Computing Systems Association). 2016. In *Proceedings of the USENIX Workshop on Cool Topics in Sustainable Data Centers*. (all published papers in the proceedings).

[25] USENIX (Advanced Computing Systems Association). 2018. In *Proceedings of the USENIX Workshop on Hot Topics in Edge Computing*. (all published papers in the proceedings).

[26] Stefano Avallone, Marcello Esposito, Antonio Pescape, Simon Pietro Romano, and Giorgio Ventre. 2002. Mtools: A one-way delay and round-trip-time meter. In *Proceedings of the Recent Advances in Computers, Computing and Communications*. WSEAS PRESS.

[27] Stefano Avallone, Salvatore Guadagno, Donato Emma, Antonio Pescape, and Giorgio Ventre. 2004. D-ITG Distributed Internet Traffic Generator. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, 2004. QEST 2004*. IEEE, 316–317.

[28] Yariv Bachar and Ophir Ovadia. 2002. NTGen Project. Retrieved 13 April, 2020 from http://softlab-pro-web.technion.ac.il/projects/NTGen/html/ntgen.htm.

[29] Bastian Ballman and Stefan Krecher. 2005. IP-Packet Generator. Retrieved 13 April, 2020 from http://p-a-t-h.sourceforge.net/html/index.php.

[30] Paul Barford and Mark Crovella. 1998. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. ACM, 151–160.

[31] David Barroso. 2020. Yersinia Traffic Generator. Retrieved 13 April, 2020 from https://github.com/tomac/yersinia.

[32] Bruno Benchimol. 2005. VoIP Traffic Generator. Retrieved 13 April, 2020 from http://voiptg.sourceforge.net/.

[33] Philippe Biondi. 2011. *Scapy*. Scapy Community. Retrieved 13 April, 2020 from https://scapy.net/.

[34] Nicola Bonelli, Stefano Giordano, Gregorio Procissi, and Raffaello Secchi. 2005. Brute: A high performance and extensible traffic generator. In *Proceedings of the SPECTS*. International Society for Modeling and Simulation, 839–845.

[35] Candela Technologies. 2020. Lanforge: Stateful IP Traffic Generators and Network Emulators. Retrieved 13 April, 2020 from http://www.candelatech.com/.

[36] Pedro Casas, Andreas Sackl, Sebastian Egger-Lampl, and Raimund Schatz. 2012. YouTube & Facebook Quality of Experience in Mobile Broadband Networks. In *Proceedings of the 2012 IEEE Globecom Workshops*. IEEE. DOI: https://doi.org/10.1109/GLOCOMW.2012.6477764

[37] Cisco. 2019. *TRex: Realistic Traffic Generator*. Cisco. Retrieved 09 October, 2020 from https://trex-tgn.cisco.com/.

[38] Cisco Inc. 2005. Using Test TCP (TTCP) to Test Throughput. Retrieved 13 April, 2020 from https://www.cisco.com/c/en/us/support/docs/dial-access/asynchronous-connections/10340-ttcp.html.

[39] Cisco Inc. 2020. Cisco Annual Internet Report, 2018–2023. *Global Mobile Data Traffic Forecast* 2020, (2020), 5. Retrieved on 8 November, 2020 from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490.html.

[40] ZTI Communications. 2020. LanTraffic V2. Retrieved 13 April, 2020 from https://www.zti-communications.com/lantrafficv2/.

[41] ITC (International Teletraffic Congress). 2011. In *Proceedings of the International Workshop on Modeling, Analysis, and Control of Complex Networks.* (all published papers in the proceedings).

[42] Universita' degli Studi di Napoli. 2020. Other Internet Traffic Generators. Retrieved 13 April, 2020 from http://www.grid.unina.it/software/ITG/link.php.

[43] Valery Diomin and Yakov Tetruashvili. 2010. Cat Karat Packet Builder. Retrieved 13 April, 2020 from https://sites.google.com/site/catkaratpacketbuilder/.

[44] Donfrays Software. 2018. Inter-Networking Test Traffic Generator. Retrieved 13 April, 2020 from http://www.donfraysoftware.com/MITS/MITS.htm.

[45] East Coast Data Comm Inc. 2019. Stateful Traffic Generator. Retrieved 13 April, 2020 from https://www.ecdata.com/Products/Stateful-Traffic-Generator/.

[46] Paris Eloy. 2018. The Network Expect Project. Retrieved 13 April, 2020 from https://www.netexpect.org/.

[47] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A scriptable high-speed packet generator. In *Proceedings of the 2015 Internet Measurement Conference.* ACM, New York, NY, 275–287. DOI: https://doi.org/10.1145/2815675.2815692

[48] Paul Emmerich, Sebastian Gallenmüller, Gianni Antichi, Andrew W. Moore, and Georg Carle. 2017. Mind the gap: A comparison of software packet generators. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems* IEEE Press, Piscataway, NJ, 191–203. DOI: https://doi.org/10.1109/ANCS.2017.32

[49] Excentis Inc. 2013. ByteBlower—Making Accurate IP Testing Quick and Easy. Retrieved 13 April, 2020 from https://www.excentis.com/products/byteblower.

[50] Wu-chang Feng, Ashvin Goel, Abdelmajid Bezzaz, Wu-chi Feng, and Jonathan Walpole. 2003. Tcpivo: A high-performance packet replay engine. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research.* ACM, 57–64.

[51] B. Fink and R. Scott. 2006. Nuttcp, v5. 3.1. Retrieved 23 Jan, 2020 from https://www.nuttcp.net/.

[52] Matias Fontanini. 2019. Libtins: C++ Packet Sniffing and Crafting Library. Retrieved 13 April, 2020 from https://libtins.github.io/.

[53] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International Conference ACM CoNEXT'10.* ACM, 12 pages.

[54] ACM (Association for Computing Machinery). 2006–2017. In *Proceedings of the ACM SIGCOMM Workshop on Mobility in the Evolving Internet Architecture (MobiArch).* (all published papers in the proceedings).

[55] ACM (Association for Computing Machinery). 2006–2018. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research.* (all published papers in the proceedings).

[56] ACM (Association for Computing Machinery). 2006–2018. In *Proceedings of the ACM SIGCOMM Workshop on Network Meets AI & ML (NetAI).* (all published papers in the proceedings).

[57] ACM (Association for Computing Machinery). 2006–2018. In *Proceedings of the ACM Workshop on Challenged Networks.* (all published papers in the proceedings).

[58] ACM (Association for Computing Machinery). 2006–2018. In *Proceedings of the Special Interest Group on Data Communication.* (all published papers in the proceedings).

[59] ACM (Association for Computing Machinery). 2008–2018. In *Proceedings of the ACM SIGCOMM Workshop on Networked Systems for Developing Regions.* (all published papers in the proceedings).

[60] ACM (Association for Computing Machinery). 2008–2018. In *Proceedings of the ACM Workshop on the Economics of Networks, Systems and Computation.* (all published papers in the proceedings).

[61] ACM (Association for Computing Machinery). 2010–2011. In *Proceedings of the ACM SIGCOMM Workshop on Home Networks.* (all published papers in the proceedings).

[62] ACM (Association for Computing Machinery). 2010–2018. In *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networks.* (all published papers in the proceedings).

[63] ACM (Association for Computing Machinery). 2011. In *Proceedings of the ACM SIGCOMM workshop on Green networking*. (all published papers in the proceedings).

[64] ACM (Association for Computing Machinery). 2011–2017. In *Proceedings of the ACM Conference on Information-Centric Networking*. (all published papers in the proceedings).

[65] ACM (Association for Computing Machinery). 2012–2014. In *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. (all published papers in the proceedings).

[66] ACM (Association for Computing Machinery). 2014–2016. In *Proceedings of the ACM Workshop on All Things Cellular*. (all published papers in the proceedings).

[67] ACM (Association for Computing Machinery). 2015. In *Proceedings of the ACM SIGCOMM Workshop on Ethics in Networked Systems Research*. (all published papers in the proceedings).

[68] ACM (Association for Computing Machinery). 2016. In *Proceedings of the ACM SIGCOMM Workshop on Fostering Latin-American Research in Data Communication*. (all published papers in the proceedings).

[69] ACM (Association for Computing Machinery). 2016–2018. In *Proceedings of the Applied Networking Research Workshop*. (all published papers in the proceedings).

[70] ACM (Association for Computing Machinery). 2017–2018. In *Proceedings of the ACM SIGCOMM 2018 Workshop on IoT Security and Privacy*. (all published papers in the proceedings).

[71] ACM (Association for Computing Machinery). 2017–2018. In *Proceedings of the ACM SIGCOMM Workshop on Mobile Edge Communications*. (all published papers in the proceedings).

[72] ACM (Association for Computing Machinery). 2017–2018. In *Proceedings of the Asia-Pacific Workshop on Networking*. (all published papers in the proceedings).

[73] V. S. Frost and B. Melamed. 1994. Traffic modeling for telecommunications networks. *IEEE Communications Magazine* 32, 3 (March 1994), 70–81. DOI: https://doi.org/10.1109/35.267444

[74] GL Communications. 2020. GL Traffic Generator: Simulation & Analysis Network Traffic Characteristics. Retrieved 13 April, 2020 from https://www.gl.com/traffic-generators.html.

[75] Eric Lee Helvey. 1998. *Trafgen: An Efficient Approach to Statistically Accurate Artificial Network Traffic Generation*. Ph. D. Dissertation. Ohio University. Retrieved 13 April, 2020 from https://etd.ohiolink.edu/pg_10?0::NO:10:P10_ACCESSION_NUM:ohiou1176494135.

[76] *Hobbit*. 1995. *Netcat*. Netcat. Retrieved from http://nc110.sourceforge.net/.

[77] Colasoft Inc. 2020. Colasoft Packet Builder - Colasoft. Retrieved 13 April, 2020 from https://www.colasoft.com/download/products/download_packet_builder.php.

[78] Rick Jones. 1996. *Netperf*. Hewlett-Packard. Retrieved 13 April, 2020 from https://github.com/HewlettPackard/netperf.

[79] Roel Jonkman. 1994. *Netspec: Philosopy, Design and Implementation*. Ph. D. Dissertation. University of Kansas, Lawrence, Kansas, USA.

[80] K. Kant, V. Tewari, and R. Iyer. 2001. Geist: A generator for E-commerce & internet server traffic. In *Proceedings of the 2001 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 49–56. DOI: https://doi.org/10.1109/ISPASS.2001.990676

[81] Stein Karyl. 1998. Spak-0.6b - Arbitrary Packet Generator/Sender. Retrieved 07 May, 2020 from http://static.lwn.net/lwn/1998/0312/a/spak.html.

[82] Konstantinos V. Katsaros, George Xylomenos, and George C. Polyzos. 2012. GlobeTraff: A traffic workload generator for the performance evaluation of future internet architectures. In *Proceedings of the 2012 5th International Conference on New Technologies, Mobility and Security*. 1–5. DOI: https://doi.org/10.1109/NTMS.2012.6208742

[83] Keysight Technologies. 2020. BreakingPoint VE - Virtualized Security Resilience Testing for Enterprise-Wide Networks. Retrieved 13 April, 2020 from https://www.ixiacom.com/products/breakingpoint-ve.

[84] Keysight Technologies. 2020. Ixchariot - Instant Performance Assessment of Complex Networks from Pre- to Post-Deployment. Retrieved 13 Sept., 2020 from https://www.ixiacom.com/products/ixchariot.

[85] Keysight Technologies. 2020. Ixnetwork - L2-3 Network Infrastructure Performance Testing That Scales to Business Needs. Retrieved 13 April, 2020 from https://www.ixiacom.com/products/ixnetwork.

[86] S. S. Kolahi, S. Narayan, D. D. T. Nguyen, and Y. Sunarto. 2011. Performance monitoring of various network traffic generators. In *Proceedings of the 2011 UkSim 13th International Conference on Computer Modelling and Simulation*. 501–506. DOI: https://doi.org/10.1109/UKSIM.2011.102

[87] G. Kramer. 2014. Generator of Self-Similar Traffic. Retrieved 13 April, 2020 from http://research.glenkramer.com/code/trf_gen3.shtml.

[88] Charles Krasic. 2002. Home Page of Mxtraf. Retrieved 13 April, 2020 from http://mxtraf.sourceforge.net/.

[89] Chia-Yu Ku, Ying-Dar Lin, Yuan-Cheng Lai, Pei-Hsuan Li, and Kate Ching-Ju Lin. 2012. Real traffic replay over wlan with environment emulation. In *Proceedings of the 2012 IEEE Wireless Communications and Networking Conference*. IEEE, 2406–2411. DOI: https://doi.org/10.1109/WCNC.2012.6214199

[90] Hakawati Security Lab. 2018. *Traffic Generators*. Hakawati Security Lab. Retrieved 13 April, 2020 from http://www.hakawati.co.kr/318.

[91] ESnet/Lawrence Berkeley National Laboratory. 2014. *iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool*. Energy Sciences Network (ESnet). Retrieved 13 April, 2020 from https://github.com/esnet/iperf.

[92] Naval Research Laboratory. 2019. *Multi-Generator (MGEN)*. U.S. Naval Research Laboratory. Retrieved from https://www.nrl.navy.mil/itd/ncs/products/mgen.

[93] Juha Laine, Sampo Saariso, and Ruii Prior. 2002. RUDE & CRUDE Traffic Generator. Retrieved 13 April, 2020 from http://rude.sourceforge.net/.

[94] Kun-Chan Lan and John Heidemann. 2002. Rapid Model Parameterization from Traffic Measurements. *ACM Transactions on Modeling and Computer Simulation* 12, 3 (2002), 201–229.

[95] LBNL/ICSI berkley lab. 2005. LBNL/LCSI Enterprise Tracing Project - Trace File Download. Retrieved 13 April, 2020 from http://www.icir.org/enterprise-tracing/download.html.

[96] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. 1994. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking* 2, 1 (Feb. 1994), 1–15. DOI: https://doi.org/10.1109/90.282603

[97] Leo Liang. 2016. IPGen IP Packets Generator. Retrieved 13 April, 2020 from https://sourceforge.net/projects/ipgen/.

[98] Jukka Manner. 2006. Jugi's Traffic Generator (jtg). Retrieved 13 April, 2020 from http://www.netlab.tkk.fi/~jmanner/jtg.html.

[99] Robert McMahon, Battu Kaushik, and Tim Auckland. 2005. *Iperf: The TCP/UDP bandwidth measurement tool*. NLANR/DAST. Retrieved 13 April, 2020 from https://sourceforge.net/projects/iperf2/.

[100] Sudhakar Mishra, Shefali Sonavane, and Anil Gupta. 2015. Study of traffic generation tools. *International Journal of Advanced Research in Computer and Communication Engineering* 4, 6 (2015), 4–7.

[101] S. Molnár, P. Megyesi, and G. Szabó. 2013. How to validate traffic generators? In *Proceedings of the 2013 IEEE International Conference on Communications Workshops*. IEEE 1340–1344. DOI: https://doi.org/10.1109/ICCW.2013.6649445

[102] David Mosberger and Tai Jin. 1998. Httperf — a Tool for Measuring Web Server Performance. *SIGMETRICS Performance Evaluation Review* 26, 3 (Dec. 1998), 31–37. DOI: https://doi.org/10.1145/306225.306235

[103] Dan Nagle. 2020. Packet Sender - Free Utility to for Sending and Receiving of Network Packets. Retrieved 13 April, 2020 from https://PacketSender.com/.

[104] Nathan Jeff. 2013. Nemesis Packet Injection Tool Suite. Retrieved 13 April, 2020 from http://nemesis.sourceforge.net/.

[105] UH Netlab. 2019–2021. *Traffic Generators Survey*. UH Netlab, University of Houston. Retrieved 13 April, 2020 from http://docs.uh-netlab.org/projects/surveypaperanalysis.

[106] Juniper Networks. 2020. WRAP17 Traffic Generator. Retrieved 13 April, 2020 from https://github.com/Juniper/warp17.

[107] NMap. 2019. Nping - Network Packet Generation Tool/Ping Utiliy. Retrieved 12 April, 2020 from https://nmap.org/nping/.

[108] University of Žilina. 2019. *Network Information Library - Traffic Generators*. Network Information Library, University of Zilina. Retrieved 12 April, 2020 from https://nil.uniza.sk/traffic-generators-list/.

[109] Robert Olsson. 2005. Pktgen the linux packet generator. In *Proceedings of the Linux Symposium*, Vol. 2, 11–24.

[110] Omnicor. 2018. Network Testing Tools. Retrieved 12 April, 2020 from http://www.omnicor.com/products/network-testing-tools.

[111] Open Source Initiative. 2020. The Open Source Definition. Retrieved 12 April, 2020 from https://opensource.org/osd.

[112] Alan Ott. 2020. PlayCap Packet Replay. Retrieved 12 April, 2020 from https://github.com/signal11/PlayCap.

[113] Srivats P. 2017. *Ostinato — Packet Generator*. Ostinato. Retrieved 12 April, 2020 from https://ostinato.org/.

[114] Pacgen Team. 2013. Pacgen Packet Generator. Retrieved 12 April, 2020 from https://sourceforge.net/projects/pacgen/.

[115] Packet Data Systems Ltd. 2019. IP-Traffic Test and Measure. Retrieved 12 April, 2020 from https://www.pds-test.co.uk/products/ip_test_measure.html.

[116] Packeth Team. 2018. packeth. Retrieved 12 April, 2020 from http://packeth.sourceforge.net/packeth/Home.html.

[117] PB Software. 2018. Network Traffic Generator and Monitor. Retrieved 12 April, 2020 from http://pbsftwr.tripod.com/id17.html.

[118] Esteban Pellegrino. 2020. pellegre/libcrafter. Retrieved 12 April, 2020 from https://github.com/pellegre/libcrafter.

[119] Postel. 2017. TG Tool. Retrieved 12 April, 2020 from http://www.postel.org/tg/.

[120] ACM SIGCOMM Conference Proceedings. 2006–2019. In *Proceedings of the Publications within the Proceedings of all ACM SIGCOMM Conferences and Journals*.

[121] USENIX Conference Proceedings and Journals. 2006–2019. In *Proceedings of the Publications within the Proceedings of all USENIX Conferences and Journals*.

[122] Vinay Ribeiro, Ryan King, and Niels Hoven. 2003. Poisson Traffic Generator. Retrieved 12 April, 2020 from http://www.spin.rice.edu/Software/poisson_gen/.

[123] Chloé Rolland, Julien Ridoux, Bruno Baynat, and Vincent Borrel. 2008. Using litGen, a realistic IP traffic model, to evaluate the impact of burstiness on performance. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST), 26.

[124] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication.* ACM, 123–137.

[125] M. Samidi. 2004. Real-Time Voice Traffic Generator. Retrieved 12 April, 2020 from http://static.lwn.net/lwn/1998/0312/a/spak.html.

[126] Henning Schulzrinne. 2017. *Traffic Generators.* Columbia University. Retrieved 12 April, 2020 from https://www.cs.columbia.edu/~hgs/internet/traffic-generator.html.

[127] Douglas C. Sicker, Paul Ohm, and Dirk Grunwald. 2007. Legal issues surrounding monitoring during network research. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement.* ACM, New York, NY, 141–148. DOI: https://doi.org/10.1145/1298306.1298307

[128] Charles Robert Simpson and George F. Riley. 2004. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *Proceedings of the Passive and Active Network Measurement.* Chadi Barakat and Ian Pratt (Eds.), Lecture Notes in Computer Science. Springer, Berlin, 168–174. DOI: https://doi.org/10.1007/978-3-540-24668-8_17

[129] Peter Siska, Marc Ph Stoecklin, Andreas Kind, and Torsten Braun. 2010. A flow trace generator using graph-based traffic classification techniques. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference.* ACM, 457–462.

[130] Skaion Corporation. 2015. Skaion Traffic Generation System (TGS). Retrieved 12 April, 2020 from http://www.skaion.com/.

[131] SolarWinds. 2020. Network Traffic Generator—WAN Killer Test. Retrieved 12 April, 2020 from https://www.solarwinds.com/engineers-toolset/use-cases/traffic-generator-wan-killer.

[132] Joel Sommers, Hyungsuk Kim, Paul Barford, and Paul Barford. 2004. Harpoon: A Flow-level Traffic Generator for Router and Network Tests. *SIGMETRICS Performance Evaluation Review* 32, 1 (June 2004), 392–392. DOI: https://doi.org/10.1145/1012888.1005733

[133] Dug Song. 2000. Fragroute. Retrieved 12 April, 2020 from https://www.monkey.org/~dugsong/fragroute/.

[134] Spirent Communications. 2020. Spirent TestCenter—Verifying Network and Cloud Evolution - Spirent. Retrieved 12 April, 2020 from https://www.spirent.com/products/testcenter.

[135] Wireshark Team. 2019. *Wireshark - Tools.* Wireshark Foundation. Retrieved 12 April, 2020 from https://wiki.wireshark.org/Tools.

[136] Qbone Testbed. 2013. Gen_send, Gen_recv: A Simple Udp Traffic Generator Application. Retrieved 12 April, 2020 from http://www.citi.umich.edu/projects/qbone/generator.html.

[137] TFGen Team. 2000. TFGen Traffic Generator. Retrieved 12 April, 2020 on http://www.pgcgi.com/hptools/.

[138] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. 2005. *Iperf: The TCP/UDP bandwidth measurement tool.* Iperf. Retrieved 12 April, 2020 from http://dast.nlanr.net/Projects.

[139] Triticom. 2006. LANDecoder32 LAN Protocol Analyzer and Traffic Monitor. Retrieved 12 April, 2020 from http://www.netunlim.com/master_site/pdfs/LD32_V3.4.pdf.

[140] Antoine Varet and Nicolas Larrieu. 2014. Realistic network traffic profile generation: Theory and practice. *Computer and Information Science* 7, 2 (2014), pp–1.

[141] Matti Vattinen. 2019. epb - Ethernet Packet Generator. Retrieved 12 April, 2020 from http://m-a-z.github.io/epb/.

[142] Kashi Venkatesh Vishwanath and Amin Vahdat. 2008. Evaluating distributed systems: Does background traffic matter? In *Proceedings of the USENIX Annual Technical Conference.* USENIX, 227–240.

[143] Kashi Venkatesh Vishwanath and Amin Vahdat. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking* 17, 3 (2009), 712–725.

[144] K. V. Vishwanath and A. Vahdat. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking* 17, 3 (June 2009), 712–725. DOI: https://doi.org/10.1109/TNET.2009.2020830

[145] Joerg Wallerich. 2008. NSWEB Traffic Generator. Retrieved 12 April, 2020 from https://www.net.t-labs.tu-berlin.de/~joerg/.

[146] Ulrich Weber. 2019. mausezahn. Retrieved 12 April, 2020 from https://github.com/uweber/mausezahn.

[147] Michele C. Weigle. 2011. Web Traffic Generation in NS2 with PackMime-HTTP. Retrieved 12 April, 2020 from https://www.cs.odu.edu/~mweigle/research/packmime/.

[148] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. 2006. Tmix: A tool for generating realistic Tcp application workloads in NS2. *SIGCOMM Computer Communication Review* 36, 3 (July 2006), 65–76. DOI: https://doi.org/10.1145/1140086.1140094

[149] Keith Wiles. 2019. The DPDK Pktgen Application - Documentation. Retrieved 12 April, 2020 from https://pktgen-dpdk.readthedocs.io/en/latest/.

[150] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. 1997. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking* 5, 1 (Feb 1997), 71–86. DOI: https://doi.org/10.1109/90.554723

[151] Tao Ye, Darryl Veitch, Gianluca Iannaccone, and S. Bhattacharya. 2005. Divide and conquer: PC-based packet trace replay at OC-48 speeds. In *Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*. IEEE, 262–271.

[152] Andy Yeow and Chin Heng. 2006. Bit-Twist: Libpcap-Based Ethernet Packet Generator. Retrieved from http://bittwist.sourceforge.net/.

[153] Petr Zach, MARTIN Pokorny, and ARNOST Motycka. 2013. Design of software network traffic generator. *Recent Advances in Circuits, Systems, Telecommunications and Control* 1, 26 (2013), 244–251.

[154] Sebastian Zander, David Kennedy, and Grenville Armitage. 2005. *Kute a High Performance Kernel-Based Udp Traffic Engine*. Technical Report 0501118A. Swinburne University of Technology. Centre for Advanced Internet Architectures (CAIA).

Received October 2020; revised June 2021; accepted September 2021