Computer Networks 56 (2012) 3531–3547

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A tool for the generation of realistic network workload for emerging networking scenarios

Alessio Botta, Alberto Dainotti *, Antonio Pescapé

University of Napoli Federico II, Department of Computer Engineering and Systems, Via Claudio 21, I-80125 Napoli, NA, Italy

ARTICLE INFO

Article history:
Received 2 August 2011
Received in revised form 8 December 2011
Accepted 26 February 2012
Available online 24 March 2012

Keywords:
Network workload
Traffic generation
Synthetic network traffic

ABSTRACT

Internet workload is a mix of many and complex sources. Therefore, its accurate and realistic replication is a difficult and challenging task. Such difficulties are exacerbated by the multidimensional heterogeneity and scale of the current Internet combined with its constant evolution. The study and generation of network workload is a moving target, both in terms of actors (devices, access networks, protocols, applications, services) and in terms of case studies (the interest expands from performance analysis to topics like network neutrality and security). In order to keep up with the new questions that arise and with the consequent new technical challenges, networking research needs to continuously update its tools. In this paper, we describe the main properties that a network workload generator should have today, and we present a tool for the generation of realistic network workload that can be used for the study of emerging networking scenarios. In particular, we discuss (i) how it tackles the main issues challenging the representative replication of network workload, and (ii) our design choices and its advanced features that make it suitable to analyze complex and emerging network scenarios. To highlight how our tool advances the state-of-the-art, we finally report some experimental results related to the study of hot topics like (a) broadband Internet performance and network neutrality violations; (b) RFC-based security and performance assessment of home network devices; (c) performance analysis of multimedia communications.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction and motivation

The Internet is today a network of incredible complexity, connecting systems that are heterogeneous for technologies and applications. Consequently, multiple types of objects and data – at different abstraction-levels – impact network workload, whose inherent complexity is further increased by its temporal evolution, coherently with the evolution of topologies, devices, network technologies, applications, and traffic. Internet workload is therefore the result of a complex mix of sources and it is quite different from the workload that was observed on large networks in the past years. For these reasons, understanding, modeling, and generating network workload are difficult and challenging tasks [1].

In this paper, we focus on the generation of *realistic workload at network level*. With the term *realistic* here we mean synthetic workload able to replicate the main (sometimes all) features of a real workload specifically targeted to a particular network scenario. Whereas the features and the configurable parameters described in Fig. 1 and in Tables 1 and 2 (further discussed in the paper) clarify the context of *workload at network level*.

Generation of network workload is a fundamental component of several networking research fields: performance of networks and network devices, including middleboxes (e.g., performance enhanced proxies, policy charging and rules function, traffic shapers), security (e.g., firewalls, intrusion and anomaly detection, background and malicious workload), quality of service and quality of

* Corresponding author.
E-mail addresses: a.botta@unina.it (A. Botta), alberto@unina.it (A. Dainotti), pescape@unina.it (A. Pescapé).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.02.019

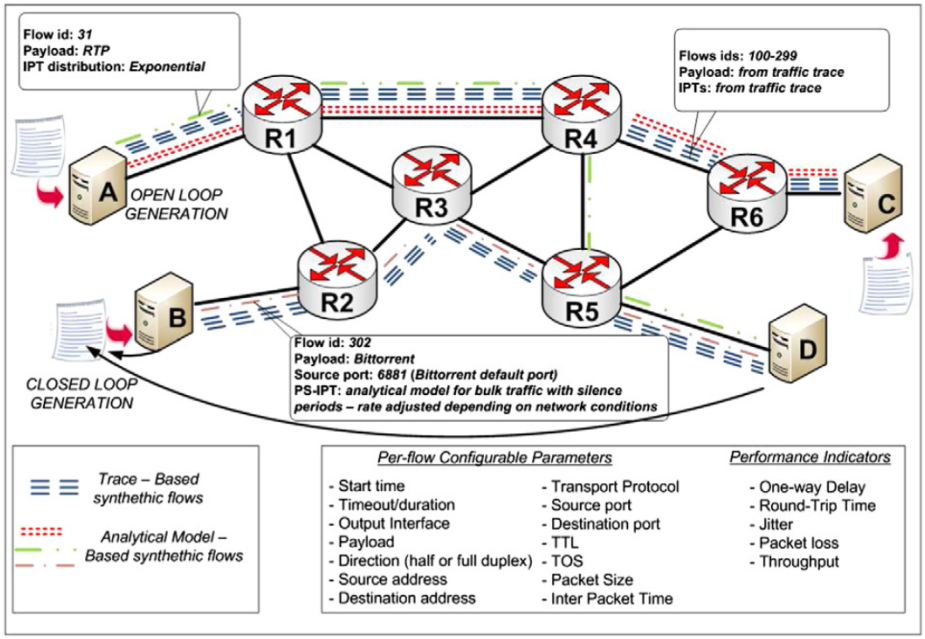


Fig. 1. Approaches and parameters for synthetic network workload generation.

Table 1
Features of our generation platform.

Computer architectures	Multiple processor architectures are supported (Strong ARM, Intel XScale, x86 32 bit and 64 bit)
Operating systems	Multiple operating systems are supported (Windows, OSX, FreeBSD, Linux, Montavista Linux, OpenWRT, snapgear)
Supported protocols	Different protocols at different layers are supported (IPv4, IPv6, ICMP, TCP, UDP, SCTP, DCCP)
Control plane	Traffic generation is driven by TSP (Traffic Specification Protocol) between sender/receivers
Centralized remote control	Sender/receivers can be remotely controlled (daemon mode) by a central manager using a separate TSP signaling channel
Distributed remote control	Sender/receivers can be remotely controlled (daemon mode) by other sender/receivers using a separate TSP signaling channel
Sending flows scalability	Each traffic sender can generate multiple parallel flows towards different receivers, using a separate thread for each flow
Receiving flows scalability	Each traffic receiver can receive multiple parallel flows from different senders, using a separate thread for each flow
Generation models	Packets can be generated with inter departure time and size emulating well known applications or following a stochastic distribution of choice
Interactions with external devices	Synchronization with external devices (e.g., high precision digital counters) on time of sent and received packets is possible through the serial ports
Metrics	Measures of throughput, jitter, losses, one way delay and round trip time with different granularities (from packet-by-packet to the entire experiment duration) can be collected
Storage of results	Detailed information is logged on sent and received packets on local or remote hosts
Bit rate and packet rate	High generation bit rate (more than 950 Mbps) and packet rate (more than 850 Kbps) can be reached over low cost COTS
Analytical model-based workload generation	Application-layer emulation of real applications (using standard Berkeley Sockets) is possible, while allowing, at the same time, the user to set several parameters at underlying levels (see Table 2)
Trace-based workload generation	Generation of realist traffic patterns can also be performed using stored pcap files Possibility to generate/receive traffic on specific addresses or interfaces

experience, new protocols (e.g., at transport-level), available bandwidth measurement, etc. Moreover, some recent networking topics (e.g. network neutrality and performance of multimedia communications (see Section 5)) can be studied only through the generation of *realistic*, *appropriate* and *configurable* network workload.

Since the generation of network workload highly depends on the layer at which the protocol stack is observed, and the analysis of its impact on networks or systems (e.g., the determination of performance indexes) often requires the replication of both its static and dynamic properties, the accurate generation of synthetic but realistic workload remains a complex and open problem. Both the literature and the market have seen the proliferation of *specific* approaches and tools purposely designed for *specific* scenarios: from the study of cluster computing, I/O, and

Table 2
Configurable parameters of our generation platform.

Host level (specified for each sender/receiver)	Log type (sent packets, received packets) and location (local or remote) Logging information inserted into the packet payload (none, minimum, extended) Interface/address on which to send/receive signaling packets
Network level (specified for each individual flow of each sender/receiver)	Interface/address on which to send/receive probing packets Time To Live Differentiated Services Byte (Type of Service)
Flow level (specified for each flow of each sender/receiver)	Duration Time to wait before start Number of packets/bytes to generate Type of application to emulate (DNS, Telnet, VoIP with different codecs, Counter Strike, Quake 3) Random distribution for the size of the packets (Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson, Gamma, Weibull) Random distribution for the inter packet time (Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson, Gamma, Weibull, On/Off) Pcap file to generate Seed of the random number generator Direction of the traffic (one way, round trip) Inter packet time recovery mechanism (to sustain the required rate in presence of frequent context switches or other disturbing factors)
Transport level (specified for each flow Stream ID for SCTP of each sender/receiver)	Source and destination ports Transport protocol (TCP, UDP, SCTP, DCCP) Nagle algorithm for TCP Congestion Control for DCCP ICMP message

multimedia, to wireless network devices, web servers and routers, and transport-level protocols. Oppositely, *general purpose* network workload generators – both software-based and hardware-based – cannot generate realistic network workload in every network scenario. The former usually run on COTS hardware and are simple in terms of functionalities and implementation, while the latter are based on custom hardware, suffer of scarce flexibility because of a closed-architecture, and are often considerably expensive. Our work starts from the assumption that approaches and systems for network workload generation are useful and effective only when they produce network workload that is realistic [2–6] and representative as much as possible of the real workload of the network scenario under study.

The contribution of this paper is twofold. First, we discuss the main features needed for, and the difficulties involved in, the generation of realistic network workload (Section 2). Then, we present a customizable tool for the generation of realistic network workload targeted to emerging networking scenarios that is based on D-ITG and its traffic generation engine [24]. More precisely, we discuss how our tool tackles the main issues challenging the representative replication of network workload (Section 3) and we illustrate its advanced features, which can be used to analyze complex and emerging network scenarios (Section 4). We highlight the design choices we made to integrate in a single and configurable platform (a *swiss-army-knife* for network workload generation) the discussed main properties, and we illustrate how the solutions we adopted tackle relevant challenges such as the support for different traffic profiles, configurability at multiple layers, scalability, repeatability of experiments, etc. Finally, to highlight how our tool advances the state-of-the-art, we show experimental results related to hot topics like broadband Internet performance and network neutrality violations, RFC-based security and performance assessment of home network devices, as well as performance analysis of multimedia communications (Section 5).

2. Realistic network workload generation

In literature a huge amount of work exists on the characterization, modeling and simulation of network workload (e.g., related to e-commerce platforms [7], live streaming media [8] and YouTube traffic [9], peer-to-peer file sharing [10], Web [11–14] and web caching [15], malicious and unwanted traffic [16–18]). Unfortunately, we cannot state the same for the generation of *realistic* network workload and for the issues associated to this important task.

In this work, we focus on the generation of *realistic* network workload over real networks and using software platforms [19,20]. Approaches for synthetic network workload generation should be able to (i) appropriately capture the complexity of real workload in different scenarios, (ii) customly alter specific properties of such workload for the purpose of the experiment, and (iii) measure indicators of the performance experienced by such workload at network level. In a novel and broader view of realistic network workload generation, towards the implementation of a *swiss-army-knife* software tool, such objectives could be achieved by combining features already present in literature in various less general approaches, plus adding new specific functionalities. Table 3 shows a (non-exhaustive) list of the platforms that are most used in literature: several powerful platforms exist, each of them with peculiar characteristics, but none of them includes the flexibility, configurability and all the features discussed in Section 3 and in Section 4.

Table 3
Other generation platforms.

OSTINATO [22]	Ostinato is an open-source, cross-platform network packet crafter/traffic generator and analyzer with a friendly GUI. It crafts and sends packets of several streams with different protocols at different rates.
SEAGULL [23]	Seagull is an open-source multi-protocol traffic generator test tool. Primarily aimed at IMS (3GPP, TISPAN, CableLabs) protocols, it is conceived for functional, load, endurance, stress and performance/benchmark tests.
Tmix [4]	Tmix is a traffic generator for ns-2 that reads a packet header trace, derives a source-level characterization of each TCP connection, and then emulates in ns-2 the application that created the TCP connections in the trace.
RUDE/CRUDE [25]	Rude/crude are two programs that can generate and collect UDP traffic. RUDE (Real-time UDP Data Emitter) is a small and flexible program that generates traffic to the network, which can be received and logged on the other side of the network with the CRUDE (Collector for RUDE).
TG [26]	TG generates and receives one-way packet traffic streams transmitted from the UNIX user level process between traffic source and traffic sink nodes in a network. In the current implementation, the TCP and UDP transport protocols, with unicast and multicast addressing (UDP only), are supported.
MGEN [27]	MGEN is a script-based tool that generates, receives and logs real-time traffic patterns. The script files can be used to emulate the traffic patterns of unicast and/or multicast UDP and TCP IP applications. MGEN currently runs on various Unix-based (including MacOS X) and Win32 platforms.
KUTE [28]	KUTE is aimed at being a maximum performance traffic generator and receiver mainly for use with Gigabit Ethernet. It is composed of Linux kernel modules (tested up to 2.6.16) which send/receive packets directly to the hardware driver (tested only with Ethernet hardware) bypassing the stack.
BRUTE [29]	BRUTE is a user space Linux application designed to produce high load of customizable IPv4 and IPv6 Ethernet traffic. The software has been designed to achieve high precision and performance in the traffic generation.
LiTGen [30]	LiTGen is on open-loop packet-level traffic generator, which statistically models IP traffic (resulting from Web requests) on a per user and application basis.
Network traffic generator [31]	Network traffic generator generates massive amounts of traffic of certain type to test network devices such as routers and firewalls.
NetSpec [32]	NetSpec provides a fairly generic framework that can be used by a user to control multiple processes across multiple hosts from a central point of control for doing network testing. NetSpec consists of daemons that implement traffic sources/sinks and various passive measurement tools.
Netperf [33]	Netperf is a benchmark that can be use to measure various aspect of networking performance. The primary foci are unidirectional data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface.
Iperf [34]	Iperf is a tool for measuring maximum TCP and UDP bandwidth performance. Iperf allows the user to tune various parameters and UDP characteristics and it reports bandwidth, delay jitter, datagram loss.
TCPivo [35]	TCPivo is a tool that provides high-speed packet replay from a trace file using standard PC hardware and freely available open-source software
TCPreplay [36]	TCPreplay has the ability to use previously captured traffic in libpcap format to test a variety of network devices.
TCPopera [37]	TCPopera has been conceived to evaluate the performance of IDSs by replaying background traffic that mimics real-world behavior.
ParaSynTG [38]	ParaSynTG is a synthetic trace generator for source-level representation of Web traffic with different characteristics such as document size and type, popularity (in terms of frequency of reference) as well as temporal locality of requests.
UniLoG [39]	UniLoG is a flexible tool to generate realistic and representative server and network loads, in terms of access requests to Web servers and creation of typical Web traffic.
Swing [40]	Swing is a closed-loop traffic generator that observes traffic on the network and extracts distributions for user, application, and network behavior. It then generates traffic corresponding to the underlying models in the ModelNet network emulation environment.
SURGE [41]	SURGE is a tool to generate Web requests according to measured statistical properties.
MACE [42]	MACE is an environment for recreating a wide range of malicious packet traffic in a laboratory testbed.

Two main alternative approaches exist in literature for the generation of network workload: (i) *trace-based* generation (TCPReplay, TCPivo, TCPopera, etc.), in which flows exactly replicate the content and the timings of traffic traces previously collected in real scenarios; (ii) *analytical model-based* generation (TG, MGEN, RUDE/CRUDE, D-ITG, etc.), in which flow and packet generation processes are based on statistical models. Generation of realistic network workload needs both approaches (as stated in [21] too, in which a simple approach is proposed), depending on the characteristics of the traffic scenario to be replicated. A tool should be able to even combine *trace-based* and *analytical model-based* techniques, that is, to generate flows with timings from a trace while filling their packets with configurable content or, viceversa, using content from the trace while inter packet times follow a custom statistical model. Fig. 1 shows the joint use of *trace-based* and *analytical-based* techniques for different traffic flows.

Most workload generators in literature work at either flow level (e.g., [20]) or packet level (e.g., [29,28]), while few of them operate instead at application level (e.g., [41]). Accurate replication of network workload requires the support of both flow-level and packet-level traffic profiles, with the additional ability to replicate specific traffic properties that are typically chosen by the application, such as, TOS fields (e.g., for experimenting with QoS), protocol ports, and transport-level payload (e.g., for experimenting with security policies, network neutrality, traffic classification). The realistic replication of some scenarios also requires the ability to manipulate headers at a higher protocol level (e.g., support to SIP, RTP). Fig. 1 shows a workload generator able to operate distributedly and to exchange between each source-sink pair several sets of traffic flows with different properties. Moreover, each source or sink is also able to measure performance indicators and to store them locally or remotely. Indeed, besides specifying the characteristics of the traffic to generate, it should be possible to collect measures on the performance experienced by the injected workload. This allows the operator to perform experiments using workload from real applications (i.e. using a *pcap* trace) and without relying on some external software (e.g., *tcpdump* + *tcptrace*) to evaluate network performance parameters such as throughput, latency, jitter, and losses. Fig. 1 contains a list of configurable parameters that it should be possible to separately set for each flow and a set of network-level performance indicators to be measured.

A workload generator should be able to operate in both *open-loop* and *closed-loop* mode [43]. In the former mode the tool works independently from the observations of the network that are carried out during the generation. In closed-loop mode, instead, the tool changes its behavior at run time according to these observations, which drive modifications to the parameters of the traffic to be generated. Automated changes range from parameters of the statistical distribution of the inter-packet time (IPT) and payload size (PS) of the packets, to the content of the packet payloads, from device (e.g., servers) log files to higher level information. Fig. 1 schematically shows how a single source may be able to generate traffic according to both *trace-based* and *analytical model-based* generation and using open-loop and closed-loop generation.

There are other requirements relevant to the realistic and accurate replication of scenarios with complex workload that should also be taken into account. Difficulties grow in – and some approaches and techniques are not applicable to – the study of large scale networks and planetary-scale testbeds (e.g., PlanetLab), while in controlled testbeds the analysis and testing phases are typically simpler [44]. Real networks have often a very large scale today, and tests have to be performed also on *large scale* in order to be representative of the reality. Large scale experiments require software architectures (hardware-based generators cannot be deployed on a large number of nodes) that are scalable and able to run on hosts that are very different, and possibly located far from each other. The support for *automated and configurable* experiments is greatly needed in such context. In large-scale networks, tests have to be performed automatically because the size of the system under test (SUT) may prevent to manually perform activities on each and every host involved in the experiment. Heterogeneous scenarios also require high configurability, because, as said, hosts may be very different from each other and may operate in significantly different conditions. Finally, the network scenarios have to be tested in the very same conditions, and therefore the workload generated needs to be *repeatable* and the obtained results *comparable*. While repeatability and comparability may be easier to achieve in the case of trace-based approaches, things change with approaches based on analytical models. In facts, the statistical distributions generated by pseudo-random number generators may be quite different from each other, especially for short time periods.

3. A synthetic workload generator

In Section 2 we reviewed the problem of the realistic replication of network workload providing our view to overcome some of the current limitations. In this section we describe the main architectural principles we adopt in the design of a network workload generator having the desirable features described in Section 2 and conceived to move a few steps towards the generation of *realistic* network workload.

Our synthetic network workload generator is based on D-ITG [24] and its traffic generation engine. The distributed (logical) architecture of our synthetic network workload generator is reported in Fig. 2. It is composed of four main modules, which can run on different hosts: **sender**/**receiver**, **logger**, **controller**, **analyzer**.

The core functionalities are provided by the module called **sender/receiver**. It can act as a traffic sender, receiver, or both, and it takes into account the aspects discussed in Section 2 (it is worth noticing that our platform can work with multiple sender/receiver instances and/or multiple senders can send to a single receiver as well as a single sender can send to multiple receivers). The measurement information can be saved directly by the sender/receivers, or it can be sent - through the network - to a module called **logger** (useful to collect all the measures

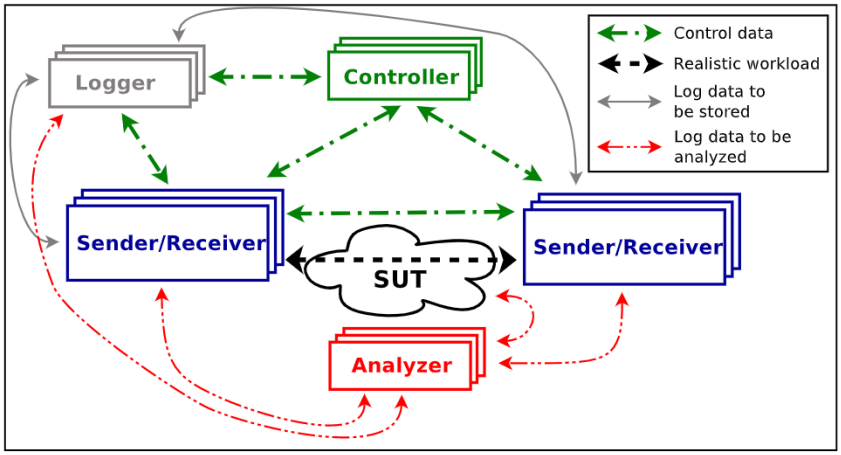


Fig. 2. Architecture of our synthetic workload generator.

in a single point or in the case of hosts with limited storage capabilities – e.g., sensors, smartphones, etc.). To cope with large scale experiments, the **sender**/**receiver** modules can be controlled directly by the user, or by another module called **controller**, which receives input from the user and interacts with the senders/receivers in order to orchestrate the measurements. This way, the user can completely control a large-scale distributed experiment from a single vantage point. A signaling channel between the main modules of the architecture (controller, sender/receiver, and logger) is used for the configuration, management and synchronization of the experiments. A module called **analyzer** is in charge of analyzing the results of the experiments – both on-line and off-line – extracting performance measures experienced by the probing traffic.

The platform is written in C and it is available at [24]. All the modules of the platform make use of multithreading. The logger and the workload senders/receivers use a single thread for the communication with the other entities (a custom protocol called Traffic Specification Protocol (TSP) has been designed for this aim) and a number of threads for their specific activities (i.e. logging or sending and receiving the workload). In more detail, they start with a single thread, which handles the communication with the other entities. Each time a request is received from the command line or from the network (e.g., the workload sender receives a request to generate a new flow from the controller), a new thread is created, which performs all the operations related to the new request (e.g., sending the packets of the flow). Workload senders and receivers use both standard and raw sockets,[1] depending on the kind of activity performed. Standard sockets are used for the communication with the other entities (i.e. the signaling), for the analytical-based generation of TCP and UDP flows, and for trace-based generation of UDP flows; raw sockets are used for ICMP flows and trace-based TCP generation. Standard sockets are used for signaling in order to take full advantage of TCP services (reliability, in order delivery, etc.) for this important task. In the case of analytical-based generation of TCP and UDP flows, standard sockets are used to experiment the same operating conditions of real network applications. For the same reason we use standard sockets to generate trace-based TCP flows. In the case of trace-based UDP flows, we use raw sockets to have full control of all the information carried by the packets at all the protocol layers. This is to accurately replicate the trace and, at the same time, to insert information required for collecting performance measures, as explained in the following. To prevent the kernel of the receiving host from generating *Destination Unreachable* ICMP messages, as instead happens in workload generators such as TCPReplay, we open standard UDP sockets also at the receiver side. However, the packets are received using raw sockets, in order to recover all the information inserted by the workload sender.

Table 1 reports the main features of our generation platform. Firstly, the platform has been developed to run on multiple hardware architectures and operating systems, which brings large flexibility and the possibility to deploy it in complex and heterogeneous scenarios (e.g., adopting also mobile platforms). This is accompanied by the support of several communication protocols at different layers, including new generation transport-level protocols as SCTP and DCCP. There are also several features particularly relevant to the orchestration and management of the experiments: (i) senders and receivers can be remotely controlled both in a centralized or distributed fashion; (ii) the data collected by these entities can be stored either locally or on a remote logger (this is particularly useful when terminals have a small amount of storage space); (iii) time synchronization with external devices is possible through serial port connections. With regard to the generation process, each sender can generate multiple parallel flows towards different receivers using a separate thread for each flow (and viceversa). The generation can either follow simple stochastic models for packet size and inter departure time, or more complex analytical models that mimic application-level protocol behavior, or instead it can follow real traffic patterns captured in traffic traces.

[1] When using standard sockets the payload of the packets is automatically encapsulated according to the transport-layer protocol in use. Raw sockets usually receive raw packets, including the transport-layer header.

During generation, several metrics can also be collected - often with packet-level granularity - as for example throughput, jitter, packet loss, one-way delay and round-trip time. Our generation platform is able to generate traffic at very high packet rates (>850 Kbps) and bit rates (>950 Mbps) using COTS Hardware (Intel Xeon E5540@2.53 GHz) and without any performance optimization; moreover, through novel software approaches as the Direct NIC Access Technology [45] that reduce packet copies in the system it is expected to make it able to operate at even higher line speeds (e.g., 10 Gbps) as done with Ostinato [22].

The configurable parameters reported in Table 2 show how the platform is highly configurable at different levels of the protocol stack. Host-level configurable parameters are specified for each sender/receiver and are mostly related to the configuration of the experiment, e.g. logging and network interfaces to be used for signaling. The interface to be used for probing/emulated traffic can also be configured, together with other parameters at network level, as Time To Live and Type Of Service. At flow level there is a rich set of parameters that can be configured: some are macro-properties of the flow (e.g., duration, time offset before start, number of packets/bytes), other parameters specify the typology of traffic to be emulated, either through statistical properties or with the pcap trace to be used as a reference, or by simply indicating the application to emulate (e.g., DNS, Telnet, VoIP and network games like Counter Strike and Quake 3). Other properties definable for each single flow include the direction of the traffic (one way or round trip) and enabling the inter packet time recovery mechanism. Finally, transport-level header fields (e.g., source and destination ports, stream ID for SCTP) and operating modes (e.g., Congestion Control for DCCP) can be configured.

The features and the related configurable parameters of our workload generator can be used to properly study current research questions. For example, the ability to collect performance measures while generating traffic that emulates real applications (e.g., in terms of IPT and PS and/or replicating portions of real traffic traces) can be used to check possible network neutrality violations (see Section 5). Also, by using various transport protocols, both traditional (UDP and TCP) and novel (SCTP and DCCP), we can test how different kinds of traffic are transported by different (and novel) network technologies, thus following the technological evolution of the networking field.

4. Advanced features introduced in the synthetic workload generator

According to the view proposed in Section 2, different issues have to be faced when actually building a software platform generating realistic workload, running on common operating systems and on COTS hardware. Our tool has been conceived and implemented carefully considering each of the issues of Section 2: it implements both *trace-based* and *analytical-based* approaches as well as their combination; it can replicate workload at different layers (e.g., packet, flow, application, etc.) and each flow is completely configurable at different layers; it works in both open-loop and closed-loop; it works distributedly (e.g., large set of instances cooperate to generate the workload) on large scale scenarios; it is configurable in an automated fashion; it guarantees repeatable and comparable experiments (see Fig. 1). In this section we describe in more detail some of the issues we coped with when building our tool, selecting those not previously considered in literature.

4.1. Synchronization among the involved entities

Often workload generators (e.g., MGEN, TCPReplay) are only able to generate traffic in one direction, i.e. from a sender to a receiver. Some others can generate traffic in both directions, but they do not synchronize the two parties (e.g., TrafGen). Real traffic is normally bidirectional, with each side of the communication affecting the other one depending on protocol interactions at different layers (e.g., packet retransmissions in TCP, or request-response communication in application-level protocols, etc.). The workload sender/receiver module of our architecture has been conceived to reproduce workload in bidirectional synchronized mode. In more detail, as regards *trace-based generation*, once two entities are provided with the same bidirectional *pcap* trace, each of them generates the packets that are related to one of the two sides of the flows inside the trace. The synchronization can be achieved in two ways: by byte-stream or timestamp. In the first case, the two entities get synchronized by looking at the sent/received bytes: each entity waits for the reception of a certain amount of data before sending its data. The amount of data to be received and sent is extracted from the packet trace. Looking at the sequence number of the TCP header, it is also possible to avoid re-injecting duplicate bytes. Using byte-stream based synchronization, the bidirectional byte-stream in the trace is exactly replicated on the network. This operating mode is therefore typically used for TCP or SCTP flows. On the other hand, the timestamp synchronization can be used for the reproducibility of non-interactive applications, as the synchronization is obtained looking at the IPT of the packets: each side sends its packets according to the timestamps in the trace. This mode is typically used for UDP or DCCP flows, and it is used in the last experiment of Section 5.

4.2. Replication of real but fully configurable packet traces

Besides the bidirectional traffic generation discussed before, *trace-based workload generation* requires the consideration of three other important aspects: how to put the data from the trace into the packets (*trace management*), how to preserve the possibility to collect measures related to the performance experienced (*measure management*), how to configure the information contained in the payload to fit the current networking scenario (*payload management*), and how to manage the timing of the packet sending (*timing management*).

Trace management must support filling the packets with the data in the pcap trace file while guaranteeing the required performance. To obtain high-performance (i.e. high generation rates), our workload generator can optionally

pre-load the whole trace in RAM before the actual generation, which avoids reading the trace from disk at generation-time (this operation obviously requires a sufficient amount of RAM on the host). The information that is actually loaded in RAM depends on the kind of generation requested. In the case of TCP, only the application layer payload is necessary, while for UDP, also the IP and UDP headers are stored in RAM. This is because for TCP we use standard sockets, while for UDP we use raw sockets, as described before. It is also worth noting that a few fields of the packet have to be modified at generation time: the identification field of the IP header and, therefore, the checksum of the transport layer header. The reason for this is described in the following.

Measure management is necessary because the generation of packets from a trace makes it difficult to collect performance measures. When performing *analytical-based generations*, our tool inserts some custom fields into the packet payload: *flow id*, *packet id*, *sending and receiving timestamps*, and *size*. These fields are dumped in the log files during the generation, in order to calculate the performance indicators for packets sent and received. In trace-based generation, the packet payload is taken from the trace and the custom fields cannot be inserted anymore. To cope with this issue, different actions are performed by our workload generator.

Firstly, most of the information (timestamps, size of the packet of UDP flows, id of packets of TCP flows, and flow id) is saved by each sending and receiving entity into a log file. After the generation these log files are parsed by the analyzer to recover the original information (i.e. the information regarding each packet sent and received).

The other pieces of information (size of the packet of TCP flows, id of the packets of UDP flows) are instead handled differently. When generating TCP flows, the actual IP packet size is decided by the TCP/IP stack of the host. TCP does not guarantee, indeed, that the bytes passed to different *send* () calls will travel through different packets (bytes are buffered and packets are sent when the kernel decides). This creates an issue in identifying the single blocks of data written by the sender into the socket buffer, possibly in different times, and therefore for determining the performance indicators for that particular data block. To cope with this issue, the receiver module is provided with the same pcap trace used by the sender, so that it knows in advance which block of data is going to be sent by the sender, and can issue a *receive* () for that particular amount of data.

When generating UDP flows, the missing piece of information is the packet id. The loss, reordering or duplication of packets, possible with this protocol, prevents to use the same operating mode described above. In this case, the packet id has to travel within the packet. Our workload generator puts this information into the identification field of the IP header. The value of this field is re-wrapped every 65,536 packets. This mechanism is therefore robust to the loss, reordering or duplication of 32,767 packets. Changing the identification field requires the re-computation of the UDP checksum. When the user requests the generation of the same trace for a number of times, this requires calculating these fields at generation time, as anticipated above.

It is worth noting that these features can be disabled in case the measure collection is not necessary.

As for *payload management*, changes may be needed to some parts of the packet payloads in order to properly replicate the scenario. This is the case, for instance, of multimedia communications using SIP. In this case, the SDP headers in the trace contain the IP addresses of the hosts involved in the original communication. In order to be realistic, such addresses have to be replaced with those of the hosts used for workload replication. Similarly, when generating RTP traffic, the timestamps in the packets have to be updated according to the current time. The sender/receiver module of our architecture can perform such trace adaptation on the fly. For flexibility and configurability, we provide a customizable configuration in which for each header field subject to change the user can specify what to do using a set of options (*set_to*, *auto_increment*, *apply_custom_transformation*, etc.). These features have been used for the last experiment of Section 5.

As for *timing management*, as described before, the choice of sending a packet can be made by either looking at the timestamp in the trace or at the bytes received from the other side. However, there is also a third possibility, to support the case in which the experimenter wants to fill the packets with the content of a packet trace and at the same time he wants to generate them with a certain pattern (i.e. timing, which also implies a certain bitrate). In this case, it is possible to mix analytical-based and trace-based generation: the packet sending times are taken from a statistical distribution (which also includes the constant distribution), while the packet content is taken from a pcap trace. This generation mode is used in the first experiment of Section 5, to emulate two different applications generating traffic at the same rate.

4.3. Replication of main states of traffic sources using tractable models

In *analytical-model based generation* – both at packet and flow level – the availability of flexible and tractable statistical models, not limited to simple marginal distributions, is an important element. For example, models should take into account mutual and temporal correlations. Moreover, useful and effective models should see the generation process of each single source as a sequence of different *states* in which the source can have very different behaviors (e.g., silent, bursty, etc.). The sender/receiver module of our architecture adopts the modeling approach presented in [46]. Specifically, customly pre-configured Hidden Markov Models (HMM) are used to reproduce the behavior of real traffic sources and then for the synthetic generation of their workload. The HMM reproduces different states in which a single source can be, and is used to associate to each state different statistical profiles of both PS and IPT, while state transitions happen with probabilities defined by a transition matrix. This can be used to accurately reproduce – for every source typology – the original workload by taking into account several statistical properties, as marginal distributions as well as mutual and temporal dependencies [46]. Fig. 3 shows an example

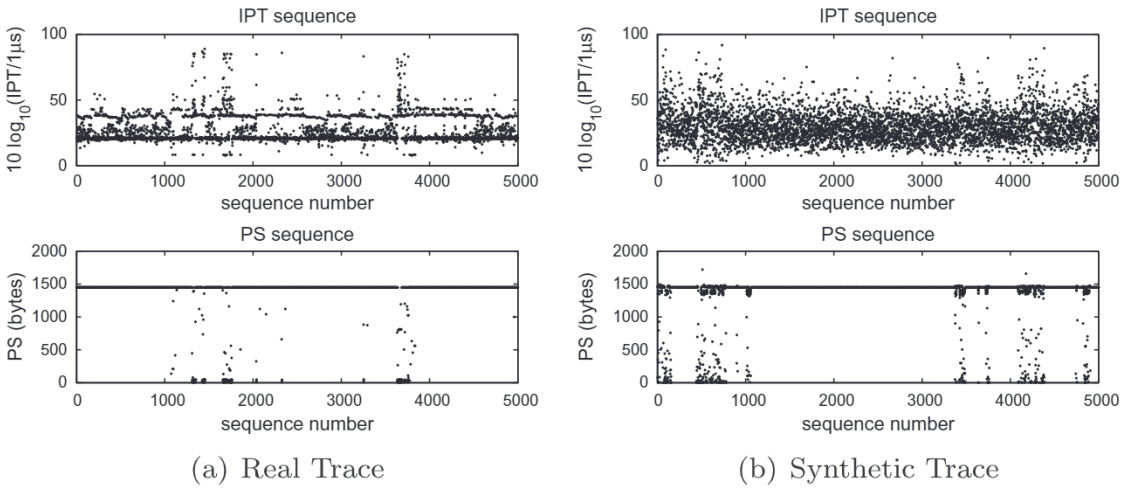


Fig. 3. HMM-based workload generation (SMTP). A logarithmic transformation is applied to the values because of the wide range of IPTs of real traffic.

in which our tool performs accurate generation/replication of time series of IPT and PS of real SMTP traffic.

4.4. Correct and accurate replication of packet timings

Accurate workload generators should guarantee the repeatability of experiments by testing the network scenarios each time in the very same conditions, and thus injecting exactly the same workload. This involves two main aspects. Firstly, when generating packets with specific IPT and PS, it should be possible (e.g., when simultaneity of packet bursts must be preserved) to set the seed for both the IPT and PS random processes. The sender/receiver of our architecture can have the seed of each random variable separately configured. Fig. 4 shows the time plot of the IPT when generating a Normal distribution ($\mu = 5$ ms and $\sigma = 30$ ms, we set IPT = 0 if IPT < 0). The left plot shows experiments performed with a workload generator in which the seed is out of the experimenter's control, while the right plot shows experiments performed with our tool in which the seed is configurable: specifying the generation seed can be used to inject into the network exactly the same traffic pattern. Secondly, to inject the very same pattern, the workload generator should be able to reproduce the IPT as accurately as possible. This important aspect, discussed in detail in [47,48], is further analyzed in the following.

Software-based synthetic workload generators can easily suffer from inaccuracies in packet sending timings – thus affecting IPT – especially when running on COTS hardware and when not altering the operating system behavior (e.g., specific drivers, realtime support in kernel scheduler, etc.). This means that even if the workload has been properly modeled or captured, its reproduction can be severely flawed by a software-based workload generator unless the timing issues are properly taken into account. For instance, Fig. 5 shows the PDF of the IPT samples obtained when generating a traffic flow with rate of 1 Mbps, a constant

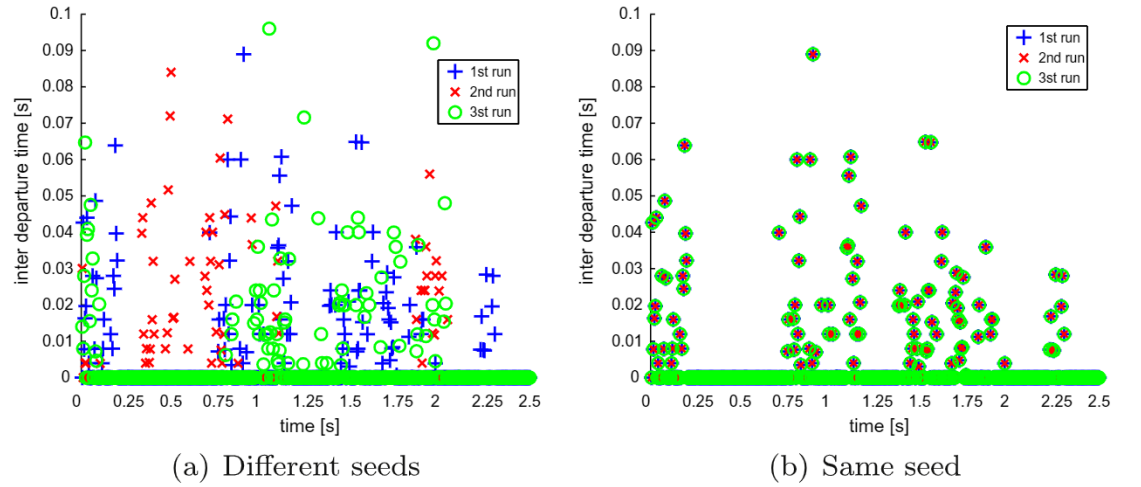


Fig. 4. IPT samples from Normal distribution.

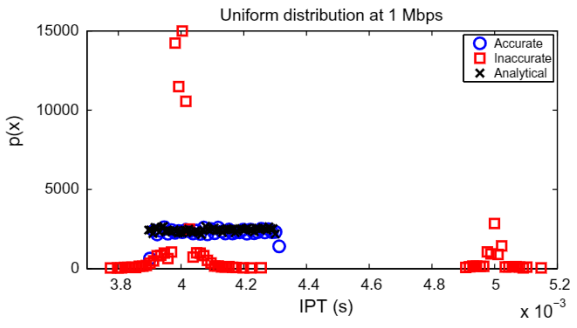


Fig. 5. IPT Samples from Uniform distribution with an accurate and non-accurate generator.

PS of 512 Bytes, and IPTs following a Uniform distribution (mean value is 4.1 ms). As a reference, in the same figure we also report the analytical Uniform distribution for such IPT (black x). The figure shows how our tool (blue[2] circle), opportunely designed and tuned, follows more strictly the analytical distribution when compared to TG [26] which does not accurately replicate the required IPT (red squares). This is because the sender/receiver of our architecture adopts the design directions presented in [47] as regards timing issues in synthetic traffic generation: (i) *polling* and (ii) *IPT recovery* in order to more accurately control actual IPTs, and (iii) buffered binary logging to reduce *internal interferences* typically happening in a software generator. We refer the reader to [47,48] for further details on the accuracy of workload generation.

It is worth noting that the IPT reported in Figs. 4 and 5 is measured on the host generating the workload. Therefore its accuracy constitutes an upper bound for the accuracy of the IPT on the network or at destination.

5. Synthetic network workload generator at work

Realistic workload generation is used in many studies, as listed in Section 1. In this section we provide some practical examples in which our platform for the generation of realistic network workload, thanks to its features, can be used for the study of hot topics in the networking field: (i) broadband Internet performance and detection of network neutrality violations; (ii) RFC-based security and performance assessment of home network devices; (iii) performance assessment of multimedia communications. The aim of this section is to provide evidences of the flexibility and usefulness of the proposed tool, while it does not intend to give deep details on each measurement scenario nor to deeply discuss the results obtained.

5.1. Broadband Internet performance and network neutrality violations

In the networking community, there is an increasing interest on both the performance of the Broadband Internet [51–54] and on the so-called "Network Neutrality" [49,50], giving rise to a lively debate involving both technical and socio-economical issues [55–58]. Network Neutrality violations are often connected to providers discriminating different network applications run by users [49,50]. For example, the provider may slow down bandwidth-hungry applications (e.g. Bittorent) rather than Web browsing, in order to avoid network saturation and improve user experience. Several techniques are available to Internet service providers for identifying network applications (a necessary step for performance discrimination), the two most common ones being based on transport-level ports and payload string matching, even if more experimental criteria can look also for other (e.g., statistical) characteristics of the flows produced by applications.

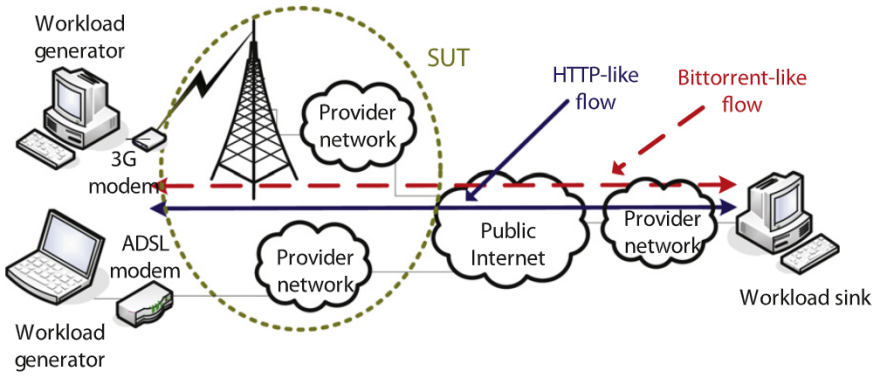
The aim of this experiment is to show how our platform can be used for investigating if and how the provider is applying some sort of discrimination to the traffic of different applications. In particular, we use the features provided by our tool – e.g., trace-based and analytical model-based generation, payload-level and transport-level configurability, active measurements, etc. – to detect the "neutrality" violations with reference to peer-to-peer traffic in two different scenarios: a residential broadband network (based on ADSL) and a mobile broadband network (based on 3G), as schematically shown in Fig. 6a.

In the first scenario, we select 4 ADSL users having the same SLA (Service Level Agreement) but with different ISPs in Italy. The considered SLA is 20 Mbps in downlink and 1 Mbps in uplink. We configure the tool to periodically generate TCP traffic flows at maximum rate (20 Mbps with maximum allowed packet size), and emulating HTTP and Bittorrent [62] in terms of transport-layer ports and packet payload. Basically, we generate two flows with the same characteristics in terms of offered bitrate and traffic profile. However, the transport-layer ports and the payloads of the packets of the two flows are taken from real traffic traces of a Bittorrent and a Web browsing session. The flows have a duration of 15 s and are not overlapped in time. Thanks to our tool, during the generation we also collect measures related to the end-to-end performance experienced by the two flows.

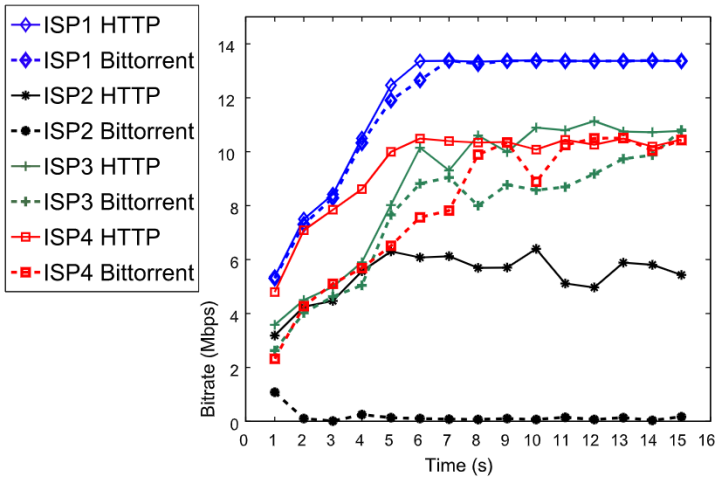
Fig. 6b shows the results obtained in downlink with TCP for both HTTP and Bittorrent and for all the ISPs (we refer to them with the terms ISP1, ISP2, ISP3 and ISP4). Looking at this figure two main considerations can be made: (i) none of the 4 ADSL reaches the bitrate of the SLA; (ii) one of the ISPs (the ISP2) out of four is actually enforcing a limitation of the Bittorrent traffic. In fact, the bitrate received when replicating the workload of this application is much lower than that received when replicating HTTP.

In the second scenario, we used the 3G network connection of one of the main operators in Italy. As a mobile user, we generate flows replicating the transport layer ports and packet payloads of Bittorrent and HTTP, but with the same characteristics in terms of offered bitrate and traffic profile. In more details, we generate through 3G connection two flows with a constant bitrate of 350 Kbps, and with packet sizes (704 Bytes) and inter-packet times (62 pps). However, as done in the previous scenario, the transport-layer ports and the payloads of the packets of the two flows are taken from real traffic traces of a Bittorrent (for the first flow) and

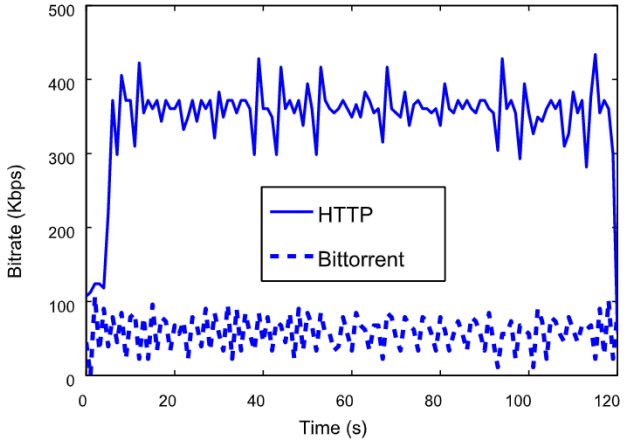
[2] For interpretation of color in Figs. 1, 2 and 4–9, the reader is referred to the web version of this article.



(a) Networks cenario



(b) Results of the experiment on the ADSL



(c) Results of the experiment on the 3G

Fig. 6. Synthetic workload generator at work for detecting network neutrality violations.

a Web browsing session (for the second one). The flows have a duration of 120 s and are not overlapped in time.

Fig. 6c shows the bitrate observed during the experiment: the performance of the Bittorrent-like flow (58 Kbps in average) is significantly lower than that of the HTTP-like flow (350 Kbps in average). This means that also this telecom operator is differentiating the service provided to different applications, reducing the bandwidth available to

Bittorrent flows. These two experiments show that by generating configurable and realistic workload and measuring the network performance it experiences, it is possible to verify that the provider is actually enforcing a policy to limit the bandwidth allocated to Bittorrent flows, and to quantitatively study its effects. Generally, in literature, this kind of analysis is performed using ad hoc tools [59].

5.2. RFC-based security and performance assessment of home network devices

Security and performance are hot topics in home networking scenarios. To study both topics in a standard, repeatable and comparable fashion, different RFCs have been proposed, addressing issues related to the assessment of the performance of network devices [60,61] and to firewall benchmarking [61]. For instance, an assessed methodology is proposed in [60] to benchmark network interconnect devices such as bridges, switches, and routers. In [60] authors state the importance of specific features that benchmarking equipment should provide: coordination between sender and receiver of probing traffic (if run on different hosts), inclusion of sequence numbers in probing packets, generation of bidirectional traffic, generation of bursty traffic (i.e. with on–off patterns), and generation of a mix of protocols approximating the conditions of a real network. Moreover, the authors stress the importance of generally accepted testing practices regarding repeatability, variance and statistical significance of small numbers of trials. In [61] different tests for firewall benchmarking are proposed: IP throughput, maximum TCP connection establishment rate, HTTP transfer rate, latency, etc. These tests require the benchmarking equipment to provide features such as: generation and recording of unidirectional and bidirectional IP flows, generation of multiple concurrent TCP connections, generation of HTTP traffic, timestamping of sent and received packets, etc. Moreover, they require the collection of different measures related to the generated traffic.

All these features are provided by our workload generator, and can be used to test home network devices for both performance and firewall capabilities, as done in this experiment. In Fig. 7a we report a high level view of the network used for the experiments. It is a home network constituted by a laptop connected to the Internet through an ADSL connection. The laptop is equipped with our workload generation platform, which communicates with a workstation located at the University of Napoli and connected to the Internet through the GARR [63]. In this simple experiment we follow the guidelines of [61]: we generate HTTP requests and responses and we measure the performance of the ADSL router/firewall with and without the firewalling features enabled. In Fig. 7b and c we respectively report some of the results regarding both firewall and baseline performance (i.e. by disabling the firewalling features on the home router). As suggested in [61] (and more precisely in its Section 5.6.5.1), such results are presented in a tabular form, showing the transfer rate achieved through the firewall when a client in the home requests multiple objects to a server outside the home using HTTP 1.1. As shown, the firewall slightly affects the performance when large objects are requested. In particular, all the HTTP requests and responses are completed with and without the firewall, and the transfer rate is about the same. More importantly, the features for bidirectional and synchronized trace replication can be used to easily generate realistic workload of real clients and servers, and to measure the performance parameters achieved by such workload.

5.3. Performance assessment of multimedia communications

Perceived quality of multimedia communications depends on performance of both signaling and data flows. Therefore, the analysis of the performance of multimedia systems has to consider issues related to both the signaling (connection set up, codec negotiation, etc.) and the multimedia flows (voice, video, etc.). To illustrate the utility of the features provided by our network workload generation tool, we show two examples in which the features introduced as well as its configurability and flexibility allow a researcher to study two interesting and emerging scenarios: multiplayer network games workload generation and analysis of the performance of real VoIP communications.

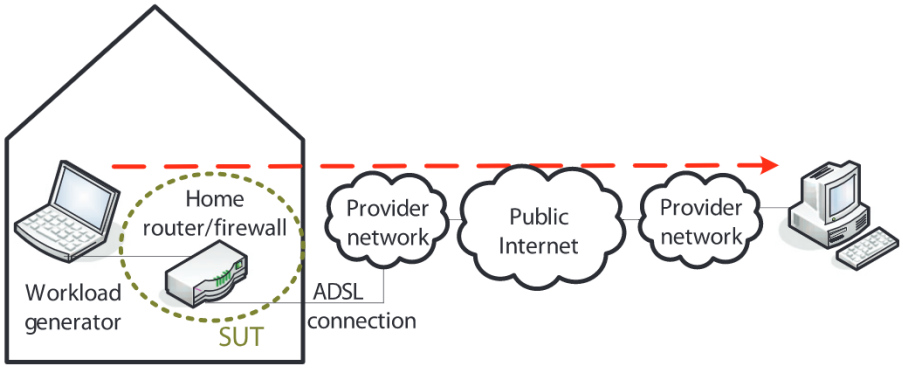
5.3.1. Multiplayer network games

Adopting the analytical model-based approach, we show how we are able to replicate the workload of a LAN party with eight Age of Mythology (AoM) players. Traffic flows generated by AoM players are captured during the LAN party. Afterwards, the traffic has been modeled in terms of IPT and PS. Such models are then added to our workload generators, and the traffic is generated over the same LAN, according to these models. In this way we replicate the same LAN party with the same number of users and with synthetic traffic. Fig. 8a shows the probability distribution functions (PDFs) of the IPT and PS of the original application traffic (Empirical), of the models (Analytical), and of the traffic generated by our tool (Synthetic). As Fig. 8a shows, the workload generator is able to correctly replicate respectively both IPT and PS of the original real traffic trace.

During the workload generation, we also record measures related to the ongoing games session and - in order to compare the bitrate achieved with real AoM players and with our tool - we also perform an experiment using our workload generator on the LAN instead of real AoM players. Fig. 8b shows the bitrate achieved in both cases: as we can simply see the real bitrate and the bitrate achieved with the workload generator are almost the same. More precisely, both experiments are 15 min long, the real traffic traces with a mean of 0.9979 Kbps and standard deviation of 0.20 Kbps and the synthetic trace with a mean of 0.9777 Kbps and a standard deviation of 0.22 Kbps. This simple experiment shows how our tool can be used to generate realistic workload of multimedia communications and to measure, at the same time, important performance parameters like bitrate.

5.3.2. VoIP applications

Adopting the trace-based approach, we test the performance of a real VoIP communication using a real applica-



(a) Network scenario

Firewall performance

Object size	100KB	1MB	5MB
Completed requests	50	50	50
	100%	100%	100%
Completed responses	50	50	50
	100%	100%	100%
Transfer Rate (bit/s)	$5.5*10^6$	$6.8*10^6$	$6.6*10^6$

(b) Results with the firewall enabled

Reference performance

Object size	100KB	1MB	5MB
Completed requests	50	50	50
	100%	100%	100%
Completed responses	50	50	50
	100%	100%	100%
Transfer Rate (bit/s)	$5.3*10^6$	$6.9*10^6$	$7.2*10^6$

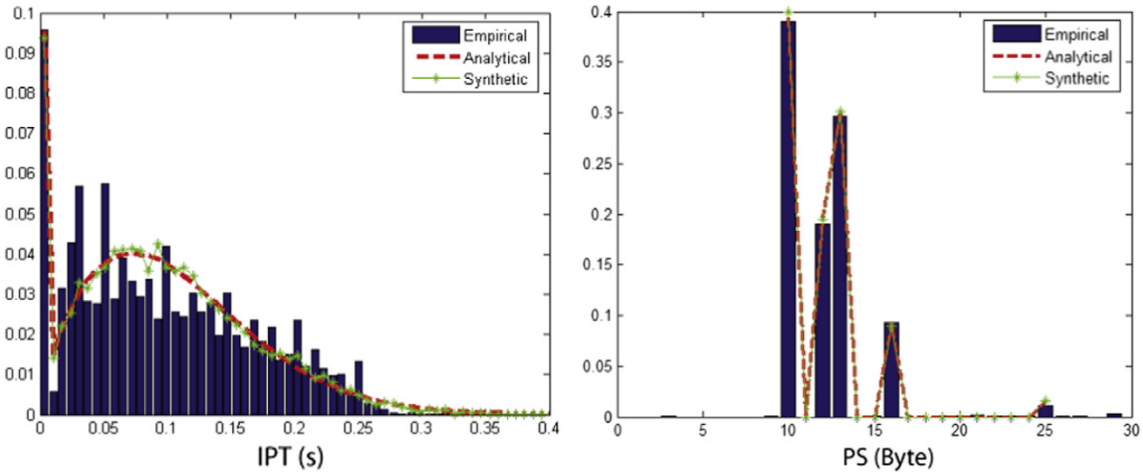
(c) Results with the firewall disabled

Fig. 7. Synthetic workload generator at work for RFC-based security and performance assessment of home network devices.

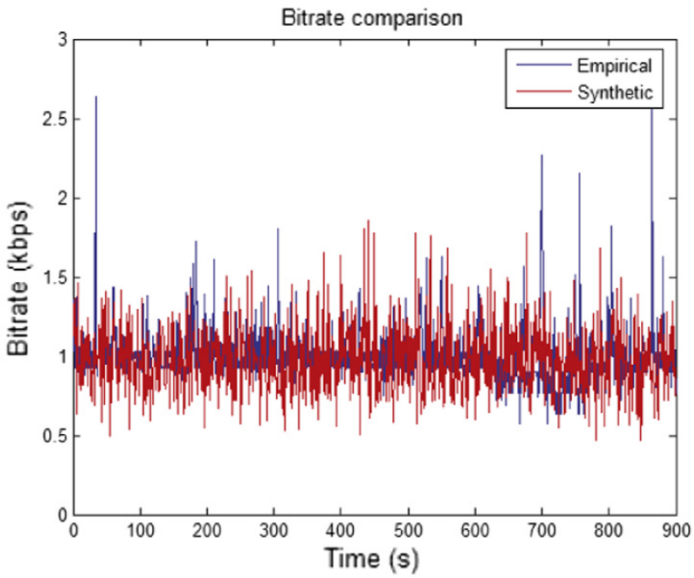
tion, namely Ekiga. Traffic flows containing SIP and RTP (audio using the *speex codec*) are captured during VoIP calls between Ekiga instances. Afterwards, the traces are replicated among our workload generators and a real Ekiga instance (Fig. 9a). The host on the left of the figure is equipped with our tool and connected to the Internet through an ADSL, while the other host is located inside our university network and it is running a real Ekiga instance. Thanks to the features for trace replication and synchronization, modifying the payload content on-the-fly, our tool is able to interact with real applications: the RTP stream is actually reproduced by Ekiga and the audio is intelligible. During the workload generation, we also record measures related to the ongoing communications. In order to measure the jitter in the two directions, we also perform an experiment using our workload generator on both sides of the network of Fig. 9a. In Fig. 9b we report the results obtained for the two multimedia flows, from a workload generator to the other and viceversa, in terms of end-to-end jitter. As shown, the jitter of both flows is lower than 1 ms, with one direction having slightly higher values due to the asymmetric bandwidth of the ADSL connection. Again, this simple experiment shows how our tool can be used to generate realistic workload of multimedia communications, interacting also with real applications, and to measure, at the same time, important performance parameters like jitter.

6. Discussion and conclusion

Network workload is the result of network events and interactions happening at different abstraction levels. Accurate modeling and generation of realistic network



(a) PDFs of Empirical vs. Analytical vs. Synthetic

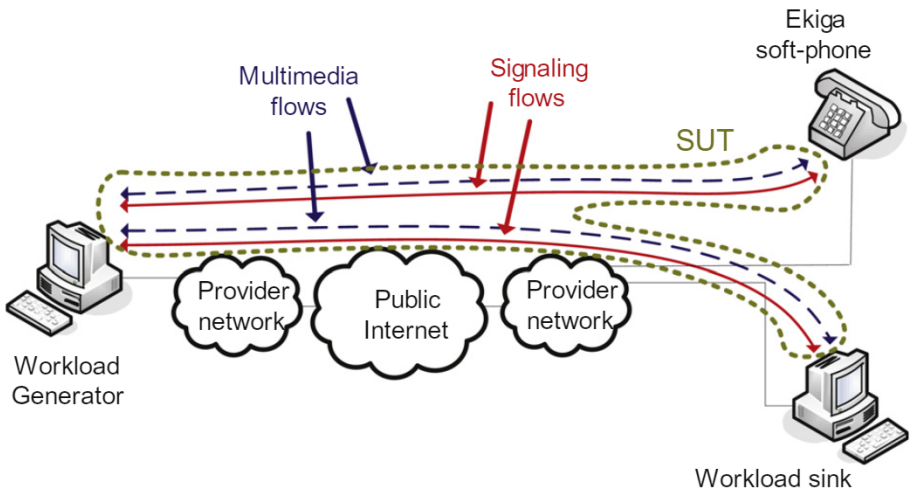


(b) Results of the experiment

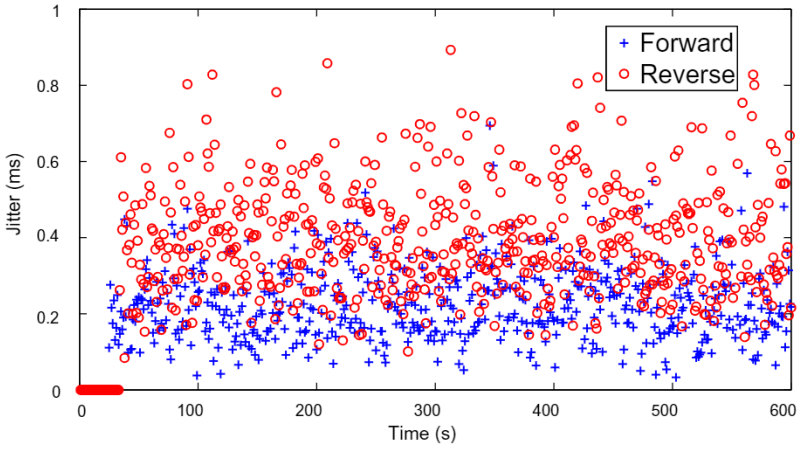
Fig. 8. Synthetic workload generator at work for AoM LAN party.

workload are difficult and challenging tasks because of heterogeneity, scale and complexity of the current Internet. In addition, the continuous temporal evolution of the network workload seriously compromises the fidelity to reality of both theoretical models and real platforms designed and developed in the past. In our view, the generation of realistic network workload (using software platforms) should reflect and take into account these considerations and should be pursued using a flexible and highly configurable approach. The intrinsic nature of *swiss-army-knife* of our tool tackles the main issues challenging this field. As clearly shown in the previous examples, our tool represents a framework for generating the requested network workload providing the possibility to (i) act as both *trace-based* and *analytical model-based* workload generator, (ii) set several parameters at several abstraction layers (packets, flows, objects, applications, users, etc.), (iii) measure performance indicators when generating synthetic workload, (iv) consider issues related to the large scale of current Internet, automated configurability of the involved entities, repeatability of the generation stages, and comparability of the results. The network scenario (number of involved hosts, emulated network scenarios, network conditions, etc.), the analytical models or the network traces, and all the other parameters of each experiment are left to the control and the decision of the users.

In this paper, we have discussed the main features that a network workload generator should have today in order to properly study current research questions and we have



(a) Network scenario



(b) Results of the experiment

Fig. 9. Synthetic workload generator at work for performance assessment of multimedia communications.

presented a tool implementing such features. The presented tool is a new development version of the D-ITG [64–68] network traffic generator. D-ITG has been used to evaluate the performance of wireless networked systems. More precisely, in [69,70] it has been used to analyze the performance of heterogeneous wireless networks, considering handoffs too. In [71] it has been used to correctly assess the capacity of wireless links. In [72,73] it has been used to evaluate the performance achieved in metropolitan wireless network in the Berlin area. In [74] it has been used to asses the impact of middleboxes on the performance of wireless networks (3G and Satellite networks). Finally, data collected using D-ITG over real heterogeneous wireless networks have been used to derive the model presented in [75]. Also, D-ITG has been used to evaluate the performance of transport protocols (e.g., SCTP) [76] and networked embedded systems (e.g., network processors) [77].

As illustrative examples, we have shown how – generating realistic network workload with the proposed tool – it is possible to practically study hot topics like broadband Internet performance and network neutrality violations, RFC-based security and performance assessment of home network devices, and performance analysis of multimedia communications. The analysis and the methodology proposed in this paper could be also used to study other hot topics like censorship [78] and the impact on the network of natural disasters [79].

Acknowledgements

We are grateful to the Area Editor and the anonymous reviewers, whose comments helped us improving the quality and the content of the paper. We thank Loreto Di Resta for helping us with the trace-based approach. This work has been partially funded by Seven One Solution srl and by LINCE project of the FARO programme jointly financed by the Compagnia di San Paolo and by the Polo delle Scienze e delle Tecnologie of the University of Napoli Federico II.

References

[1] M. Calzarossa, G. Serazzi, Workload characterization: a survey, Proceedings of the IEEE 81 (8) (1993) 1136–1150.
[2] D. Ferrari, On the foundation of artificial workload design, in: Proc. ACM SIGMETRICS, 1984, pp. 8–14.
[3] Orabi Shurrab, Jules Pagna Disso, Irfan Awan, A realistic approach to background traffic generator, in: 27th Annual UK Performance Engineering Workshop, 7–8 July 2011, 2011.
[4] M.C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, F.D. Smith, Tmix: a tool for generating realistic TCP application workloads in ns-2, ACM SIGCOMM Computer Communication Review (CCR) 36 (3) (2006) 67–76.
[5] P. Barford, M. Crovella, Generating representative Web workloads for network and server performance evaluation, SIGMETRICS – Performance Evaluation Review 26 (1) (1998) 151–160.
[6] Y. Choi, J.A. Silvester, H. Kim, Analyzing and modeling workload characteristics in a multiservice IP network, IEEE Internet Computing (2011) 35–42.
[7] D.A. Menasce, Workload characterization, IEEE Internet Computing 7 (5) (2003) 89–92.
[8] E. Veloso, V. Almeida, W. Meira Jr., A. Bestavros, J. Shudong, A hierarchical characterization of a live streaming media workload, IEEE/ACM Transactions on Networking 14 (1) (2006) 133–146.
[9] Abdolreza Abhari, Mojgan Soraya, Workload generation for YouTube, Multimedia Tools Applications Journal 46 (1) (2010) 91–118.
[10] P.K. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, J. Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: SOSP, 2003, pp. 314–329.
[11] M. Arlitt, C. Williamson, Internet web servers: workload characterization and performance implications, IEEE/ACM Transactions on Networking 5 (5) (1997) 631–645.
[12] A. Williams, M. Arlitt, C. Williamson, K. Barker, Web Workload Characterization: Ten Years Later, Springer, Heidelberg, 2005.
[13] R. Hashemian, D. Krishnamurthy, M. Arlitt, Web workload generation challenges – an empirical investigation, Software: Practice and Experience (2011).
[14] R. Pena-Ortiz et al., Dweb model: representing Web 2.0 dynamism, Computer Communications 32 (6) (2009) 1118–1128.
[15] Mudashiru Busari, Carey Williamson, ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches, Computer Networks 38 (6) (2002) 779–794. 22 April.
[16] F. Ricciato, Unwanted traffic in 3G networks, Computer Communication Review 36 (2) (2006) 53–56.
[17] F. Ricciato, E. Hasenleithner, P. Romirer-Maierhofer, Traffic analysis at short time-scales: an empirical case study from a 3G cellular network, IEEE Transactions on Network and Service Management 5 (1) (2008) 11–21.
[18] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, G. Huston, Internet background radiation revisited, IMC (2010) 62–74.
[19] K.V. Vishwanath, A. Vahdat, Swing: realistic and responsive network traffic generation, IEEE/ACM Transactions on Networking 17 (3) (2009) 712–725.
[20] J. Sommers, H. Kim, P. Barford, Harpoon: a flow-level traffic generator for router and network tests, SIGMETRICS – Performance Evaluation Review 32 (1) (2004). 392-392.
[21] Christopher Mueller, Michael Horie, Stephen W. Neville, A distributed application-level IT system workload generator, in: International Conference on Advanced Information Networking and Applications (AINA '09), 2009.
[22] http://code.google.com/p/ostinato/.
[23] http://gull.sourceforge.net/#Open+Source.
[24] http://www.grid.unina.it/software/ITG/.
[25] http://rude.sourceforge.net/.
[26] http://www.postel.org/tg/tg.html.
[27] http://cs.itd.nrl.navy.mil/work/mgen/.
[28] http://caia.swin.edu.au/genius/tools/kute/.
[29] http://code.google.com/p/brute/.
[30] C. Rolland, J. Ridoux, B. Baynat, Catching IP traffic burstiness with a lightweight generator, in: Proc. IFIP NETWORKING'07, Atlanta, May 14–18, 2007, pp. 924–934.
[31] http://sourceforge.net/projects/traffic/.
[32] http://www.ittc.ku.edu/netspec/.
[33] http://www.netperf.org/netperf/.
[34] http://sourceforge.net/projects/iperf/.
[35] http://www.thefengs.com/wuchang/work/tcpivo/.
[36] http://tcpreplay.synfin.net/.
[37] http://wwwcsif.cs.ucdavis.edu/wongfs/research.html.
[38] R. El Abdouni Khayari, M. Rucker, A. Lehmann, A. Musovic, ParaSynTG: a parameterized synthetic trace generator for representation of WWW traffic, in: Proc. of SPECTS'08, Edinburgh, June 16–18, 2008, pp. 317–323.
[39] A.W. Kolesnikov, B.E. Wolfinger, N. Luttenberger, H. Peters, Web workload generation according to the UniLoG approach, in: 17th GI/ITG Conference on Communication in Distributed Systems (KiVS 2011), 2011.
[40] http://cseweb.ucsd.edu/kvishwanath/Swing/.
[41] http://www.cs.bu.edu/crovella/links.html.
[42] Joel Sommers, Vinod Yegneswaran, Paul Barford, A framework for malicious workload generation, in: Internet Measurement Conference, 2004, pp. 82–87.
[43] Bianca Schroeder, Adam Wierman, Mor H. Balter, Open versus closed: a cautionary tale, in: NSDI 2006, vol. 3, Berkeley, CA, USA, 2006.
[44] E.A. Bretz, Network test starts in the lab, IEEE Spectrum 38 (1) (2001) 77–78.
[45] http://www.ntop.org/products/pf_ring/dna/.
[46] A. Dainotti, A. Pescapé, P. Salvo Rossi, F. Palmieri, G. Ventre, Internet traffic modeling by means of hidden Markov models, Computer Networks 52 (14) (2008) 2645–2662.
[47] A. Botta, A. Dainotti, A. Pescapé, Do you trust your software-based traffic generator?, IEEE Communications Magazine 48 (9) (2010) 158–165
[48] A. Botta, A. Dainotti, A. Pescapé, Do you know what you are generating? Co-Next 2007 Student Workshop, 2007, pp. 32.
[49] P. Kanuparthy, C. Dovrolis, DiffProbe: Detecting ISP service discrimination, in: Proc. of the IEEE INFOCOM Conference, March 2010.
[50] M. Dischinger, A. Mislove, A. Haeberlen, K. Gummadi, Detecting Bittorrent blocking, in: ACM SIGCOMM IMC, 2008.
[51] M. Dischinger, A. Haeberlen, K.P. Gummadi, S. Saroiu, Characterizing residential broadband networks, in: Proc. ACM SIGCOMM Internet Measurement Conference, San Diego, CA, USA, October 2007.
[52] G. Bernardi, M.K. Marina, Bsense: a system for enabling automated broadband census: short paper, in: Proc. of the 4th ACM Workshop on Networked Systems for Developing Regions (NSDR '10), June 2010, 2010.
[53] C. Kreibich, N. Weaver, B. Nechaev, V. Paxson, Netalyzr: illuminating the edge network, in: Proc. Internet Measurement Conference, Melbourne, Australia, November 2010.
[54] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, Antonio Pescapé, Broadband internet performance: a view from the gateway, in: Proceedings of the ACM SIGCOMM 2011 Conference, New York, NY, USA, 2011, pp. 134–145.
[55] NationalBroadbandPlan, http://www.broadband.gov/.
[56] http://www.broadband-europe.eu/.
[57] D.J. Weitzner, Net neutrality... seriously this time, IEEE Internet Computing 12 (3) (2008) 86–89.
[58] G. Goth, The global net neutrality debate: back to square one? IEEE Internet Computing 14 (4) (2010) 7–9..
[59] M. Dischinger, M. Marcon, S. Guha, K.P. Gummadi, R. Mahajan, S. Saroiu, Glasnost: enabling end users to detect traffic differentiation, in: USENIX NSDI'10, 2010.
[60] S. Bradner, J. McQuaid, RFC 2544: benchmarking methodology for network interconnect devices, in: IETF, 1999.
[61] B. Hickman, D. Newman, S. Tadjudin, T. Martin, RFC 3511: benchmarking methodology for firewall performance, in: IETF, 2003.
[62] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. Felber, A. Al Hamra, L. Garces-Erice, Dissecting BitTorrent: five months in a Torrent's lifetime, in: Passive & Active Measurement 2004, April 2004.
[63] http://www.garr.it/.
[64] A. Botta, A. Dainotti, A. Pescapé, Multi-protocol and multi-platform traffic generation and measurement, in: IEEE INFOCOM 2007 DEMO Session, Anchorage, Alaska, USA, May 2007.
[65] S. Avallone, D. Emma, S. Guadagno, A. Pescapé, G. Ventre, D-ITG: distributed internet traffic generator, in: First IEEE International Conference on Quantitative Evaluation of Systems (QEST 2004), Enschede, Netherlands, September 27–30, 2004.
[66] D. Emma, A. Pescapé, G. Ventre, Analysis and experimentation of an open distributed platform for synthetic traffic generation, in: 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004), Suzhou, China, May 2004, pp. 277–283, ISBN: 0-7695-2118-5.
[67] S. Avallone, D. Emma, A. Pescapé, G. Ventre, A distributed multiplatform architecture for traffic generation, in: International Symposium on Performance Evaluation of Computer and

Telecommunication Systems, (SPECTS) 2004, San Jose, California (USA), July 25–29, 2004, pp. 659–670, ISBN: 1-56555-248-9.
[68] S. Avallone, A. Pescapé, G. Ventre, Distributed Internet traffic generator (D-ITG): analysis and experimentation over heterogeneous networks, in: International Conference on Network Protocols, ICNP 2003, Atlanta–Georgia, USA, November 2003.
[69] M. Bernaschi, F. Cacace, A. Pescapé, S. Za, Analysis and experimentation over heterogeneous wireless networks, in: First IEEE International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05), Trento, Italy, February 2005.
[70] G. Iannello, A. Pescapé, G. Ventre, L. Vollero, Experimental analysis of heterogeneous wireless networks, in: WWIC 2004, Wired/Wireless Internet Communications 2004, LNCS, vol. 2957, Frankfurt, Germany, February 2004, pp. 153–164, ISBN: 3-540-20954-9.
[71] L. Angrisani, A. Botta, A. Pescapé, M. Vadursi, Measuring wireless links capacity, in: IEEE 1st International Symposium on Wireless Pervasive Computing, 2006, 16–18 January 2006, pp. 1–5.
[72] Roger Karrer, Istvan Matyasovszki, Alessio Botta, Antonio Pescapé, Experimental evaluation and characterization of the magnets wireless backbone, in: ACM WINTECH, 2006, pp. 26–33.
[73] R. Karrer, I. Matyasovszki, A. Botta, A. Pescapé, MagNets: experiences from deploying a joint research-operational next-generation wireless access network testbed, in: Proc. of 3rd International Conference on Testbeds and Research Infrastructures (TridentCom), Orlando, FL, May 2007.
[74] A. Botta, A. Pescapé, Monitoring and measuring wireless network performance in the presence of middleboxes, in: The 8th International Conference on Wireless On-demand Network Systems and Services (WONS), Bardonecchia (TO), Italy, January 2010.
[75] G. Iannello, F. Palmieri, A. Pescapé, P. Salvo Rossi, end-to-end packet-channel Bayesian model applied to heterogeneous wireless networks, in: IEEE Globecom 2005 General Conference, St. Louis, MO, USA, December 2005, ISBN: 0-7803-9415-1.
[76] A. Dainotti, S. Loreto, A. Pescapé, G. Ventre, SCTP performance evaluation over heterogeneous networks, Concurrency and Computation: Practice and Experience (Wiley) 19 (8) (2007). Special Issue on Performance Analysis and Enhancements of Wireless Networks, pp. 1207–1218.
[77] A. Botta, W. de Donato, A. Pescapé, G. Ventre, Networked embedded systems: a quantitative performance comparison, in: IEEE Globecom 2008, New Orleans, LA, USA, 30 November–4 December, 2008.
[78] A. Dainotti, C. Squarcella, E. Aben, K.C. Claffy, M. Chiesa, M. Russo, A. Pescapé, Analysis of country-wide internet outages caused by censorship, in: ACM SIGCOMM Conference on Internet Measurement Conference (IMC '11), ACM, NY, USA, 2011, pp. 1–18.
[79] Z.S. Bischof, J.S. Otto, F.E. Bustamante, Distributed systems and natural disasters: BitTorrent as a global witness, in: Special Workshop on Internet and Disasters (SWID '11), ACM, NY, USA, 2011, Article 4, 8 pp.

Alessio Botta is a postdoc at the Department of Computer Engineering and Systems of the University of Napoli Federico II (Italy). He graduated in Telecommunications Engineering (M.S.) and obtained the Ph.D. in Computer Engineering and Systems, both at University of Napoli Federico II. His research interests are in the area of networking and, in particular, in the area of network performance measurement and improvement, with a specific focus on wireless and heterogeneous systems. Alessio Botta has coauthored more than 40 international journal (IEEE Communications Magazine, IEEE Transactions on Parallel and Distributed Systems, Elsevier Computer Networks, etc.) and conference (IEEE Globecom, IEEE ICC, IEEE ISCC, etc.) publications. He has served and serves several technical program committees of several international conferences (IEEE Globecom, IEEE ICC, etc.) and he acts as reviewer for different international conferences (IEEE Infocom, etc.) and journals (IEEE Transactions on Mobile Computing, IEEE Network, IEEE Transactions on Vehicular Technology, etc.) in the area of networking. In 2010 he was awarded with the best local paper award at IEEE ISCC 2010.

Alberto Dainotti received the M.S. Laurea Degree in Computer Engineering in 2004 at the University of Napoli Federico II (Italy). In 2008 he received the Ph.D. in Computer Engineering and Systems from the same university. From July 2007 to February 2008 he visited the Cooperative Association for Internet Data Analysis (CAIDA) at University of California San Diego, working on traffic classification and analysis of malware traffic. Currently he works as a Post-Doc at Department of Computer Engineering and Systems of the University of Napoli Federico II. His research interests fall in the areas of network measurements, traffic analysis, and network security. He is a member of the IEEE.

Antonio Pescapé is an Assistant Professor at the Department of Computer Engineering and Systems of the University of Napoli Federico II (Italy). He received the M.S. Laurea Degree in Computer Engineering and the Ph.D. in Computer Engineering and Systems, both at University of Napoli Federico II. His research interests are in the networking field with focus on Internet Monitoring, Measurements and Management and Network Security. Antonio Pescapé has coauthored over 90 journal (IEEE Communications Magazine, JSAC, IEEE Wireless Communications Magazine, IEEE Networks, etc.) and conference (SIGCOMM, IMC, PAM, Globecom, ICC, etc.) publications and he is co-author of several patents pending. He has served and serves on more than 70 technical program committees of IEEE and ACM conferences. He has served as Editorial Board Member of IEEE Survey and Tutorials (2007–2010) and was guest editor for the special issue of Computer Networks on "Traffic classification and its applications to modern networks". In 2009 he was awarded with the IET Communications Premium Award 2009, in 2010 he was awarded with the best local paper award at IEEE ISCC 2010, in 2011 he was awarded with the TEA (Technologybiz Endorsement Award). He is a Senior Member of the IEEE.