Scalable software architecture for distributed MMORPG traffic generation based on integration of UrBBaN-Gen and IMUNES

Valter Vasić, Mirko Sužnjević, Miljenko Mikuc, and Maja Matijašević

***Abstract*—We present a scalable software architecture for distributed traffic generation capable of producing Massively Multiplayer Online Role-Playing Game (MMORPG) packet flows in a statistically accurate manner for thousands of concurrent players. The main challenge, to achieve truly massive scale traffic generation, has been achieved by introducing kernel based virtualization, pioneered by the network simulator/emulator IMUNES, into the User Behaviour Based Network Traffic Generation (UrBBan-Gen, introduced in our earlier work). The UrBBan-Gen software architecture consists of four modules: *Service repository*, *Control function and user interface*, *Behaviour process*, and *Traffic generation process*. IMUNES has been integrated into the virtualization part of the *Traffic generation process*, which has resulted in two improvements: 1) increasing the number of generated packet flows while accurately replicating the required statistical properties, and, 2) introducing the ability to run various network scenarios in simulated, as well as real networks, under realistic traffic loads. With respect to the traffic generation capabilities of the previous version of UrBBan-Gen, which was based on Linux containers, the IMUNES based solution demonstrates higher scalability, lower packet loss rates, and lower CPU load for both the UDP traffic at high packet rate and "thin" TCP traffic flows typical for MMORPGs.**

I. Introduction

In recent years Massively Multiplayer Online Role-Playing Games (MMORPGs) have become a growing phenomenon, attracting millions of players. Networked games generate traffic which is very demanding in terms of Quality of Service (QoS), due to the real-time player interaction in MMORPG virtual worlds [1]. As the number of games and users grows, so does the amount of traffic which needs to traverse the network. According to the Cisco Visual Networking Index [2], gaming traffic is expected to grow with a compound annual growth rate of 43% in the period 2010–2015, this being the second largest growth after the video category.

In order to achieve a satisfying level of QoS for a MMORPG on a network level, a network provider must primarily ensure that the latency and jitter values are very low, and that crucial data is delivered in a timely manner. The network enabling connectivity for games needs to be well designed and dimensioned, and the network equipment must be tested under realistic IP traffic loads. The most popular MMORPGs, such as *World of Warcraft* (WoW) by *Activision Blizzard* have millions of players. For scalability reasons, the player population is distributed among multiple servers, or "shards", which replicate the entire virtual world content but limit the number of concurrent players. While the exact number of active players per shard is a matter of game content and design, it can easily be in the range of thousands or even tens of thousands, and the game traffic flow characteristics are typically highly variable and dependent on the player behaviour patterns in the game world.

Motivated by the challenge to achieve truly massive scale software traffic generation, corresponding to real MMORPG network traffic, in our previous work [3] we have developed a software architecture for distributed traffic generation based on player behaviour named User Behaviour Based Network Traffic Generator (UrBBaN-Gen). The goal of UrBBaN-Gen is to accurately generate the packet flows corresponding to thousands of concurrent players typically present in a single shard, while maintaining the statistical properties of real MMORPG traffic.

Following the initial version of UrBBaN-Gen, we introduced a scalability improvement, described in more detail in [4], by substituting the Linux Containers (LXC) virtualization technology by kernel based virtualization in IMUNES (Integrated Multiprotocol Network Emulator/Simulator) [5], [6]. The work presented in this paper goes a step further by presenting a detailed view of the software architecture of UrBBaN-Gen and IMUNES, means of their integration and achieved performance evaluation. We run additional measurements to demonstrate that the new setup is capable of synthetic generation of MMORPG traffic at a larger scale. We demonstrate the gains through measurements focusing on generated packet load and bandwidth, and the CPU load and RAM usage on a PC running the traffic generation. We also confirm that the integration did not deteriorate the generated traffic characteristics with respect to the required statistic traffic models. This integration also enables the testing of traffic workloads on realistic network topologies, thus enabling more precise and detailed tests which can be used to identify problems and bottlenecks in the network.

Following this Introduction, in Section 2 we present the related work in the area of behaviour based traffic generation with emphasis on networked games and virtualization techniques, and in Section 3 we describe the UrBBaN-Gen.

Manuscript received November 13, 2012; revised December 26, 2012. The material in this paper was presented in part at the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2012), Split, Croatia, Sept. 11-13, 2012.

Authors are with University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia (e-mails: {fvalter.vasic, mirko.suznjevic, miljenko.mikuc, maja.matijasevic}@fer.hr).

In Section 4 we explain the improvements of the emulated environment. Section 5 presents the testing methodology, and Section 6 shows the results. Section 7 concludes the paper.

II. Related work

Network traffic generators may be categorized based on how they derive the packet flow patterns, as follows:

- traffic generators based on replay of previously captured packet flows,
- traffic generators emulating the properties of previously captured traffic based on statistical analysis of the packet trace,
- source based traffic generators, which emulate the behaviour of the traffic source (user and application).

In this paper, we focus on source based traffic generators.

One of the first source level traffic generators was Scalable URL Reference Generator (SURGE) [7]. SURGE creates a realistic web workload that mimics a set of real users accessing a server. SURGE defines a concept of user equivalent (UE) as a process in an endless loop that alternates between making requests for web files, and being idle. The statistical properties of web reference streams that are needed by each UE are described by parameters such as file sizes, request sizes, popularity, embedded references, temporal locality, OFF times.

Another example is GenSyn, a synthetic traffic generator implemented in Java based on user behaviour [8]. The stochastic user behaviour is described by state diagrams. The stochastic user behaviour model controls the creation of TCP connections and UDP streams through interface modules that links the GenSyn process to the underlying Internet protocol stack on the workstation.

To generate MMORPG traffic a model must be created and implemented in a specific traffic generator. While there are several models of network traffic of MMORPGs described in literature [9], [10], [11] there are very few implementations of models in traffic generators. The extensive overview of research efforts in areas of traffic analysis and modelling in MMORPGs can be found in [12]. Some works on MMORPG traffic focus on simulation in NS2 [9], [13], [14]. Simulation of MMORPG traffic of different behaviours and different access networks is presented in [13], while influence of different TCP versions on coexistence of MMORPG and FTP traffic is investigated in [14]. Shin et al. [15] propose a novel method for modelling the network traffic of games. They analyze packet size and inter-arrival times of WoW and first person shooter game *Left 4 Dead* (L4D) by *Turtle Rock Studios*. Authors propose a transformational scheme in order to simplify the shape of the traffic so it can be mapped to an analytical model. They implement their model in an online game traffic generator [16]. Authors claim that their traffic model is based on player behaviour, but this behaviour is only referred to as high erraticism of the traffic. On the other hand, traffic generation process used in UrBBaN-Gen [3] is fully defined by application level user behaviour similarly to GenSyn. In the area of other gaming traffic and traffic of machine to machine (M2M) applications, there is a mobile application that has been developed within the FP7 LOLA project [17].

With a working traffic generator model, a virtualized emulator is needed to deploy a sufficient number of network nodes that will run traffic generators. There are a couple of integrated network topology emulator solutions that use kernel based virtualization. The first emulator to use this kind of technology was IMUNES [5], [6]. IMUNES runs natively on FreeBSD, and it can also run in virtual machine software such as VMware. The FreeBSD kernel has recently been updated with a better routing table lookup algorithm capable of achieving 490 million lookups per second in synthetic tests using uniformly random IPv4 keys on a commodity 8-core CPU [18]. Since IMUNES runs on the top of FreeBSD kernel this performance can be used by virtual nodes in IMUNES. IMUNES has also served as a foundation of the CORE network emulator [19]. It is fully based on an older IMUNES release with improvements regarding mobility. It has been ported to work on Linux distributions and no longer works on the newer FreeBSD releases. A similar network emulator is also mininet [20] which provides a much simpler GUI and also runs on Linux. IMUNES was chosen because of stability, performance, user-friendliness and GUI, advanced scripting options, and capabilities for testbed automation.

III. Introduction to UrBBaN-Gen

The goal of UrBBaN-Gen is to generate realistic client and server traffic of complex IP services, based on user behaviour, described through a mathematical model which captures the statistical characteristics of the user behaviour and the generated traffic. A source based traffic model for MMORPGs is based on player behaviour on the application level, and defined through action categories [21] and the respective traffic models for each action category [22].

A. Supported services

While in our work UrBBan-Gen has primarily used MMORPG statistical models, its functional architecture and implementation are service independent, so any new services may be added through new user behaviour and traffic models. The design is highly modular, enabling the design and development of specific extensions. Both main modules, described later in more detail, have been developed and tested separately (the behaviour simulation [21] and the traffic modelling and generation [22]). Each module has defined outputs so newly implemented models can be tested and validated separately. An MMORPG provider could use UrBBaN-Gen to test their network under normal operating conditions, but could also easily create massive "stress tests". The traffic generation capacity is expandable by possible replicating the UrBBaN-Gen on multiple PCs, depending on the required scale of traffic generation. The ability to control and achieve a very large scale is very important as users' interest in MMORPGs is highly correlated to the game content releases – when the new game content is released, the servers can experience the loads several times higher than the average load.

As a case study of a complex real time service we used MMORPGs as they involve large number of users with diverse application level behaviours which significantly affect network

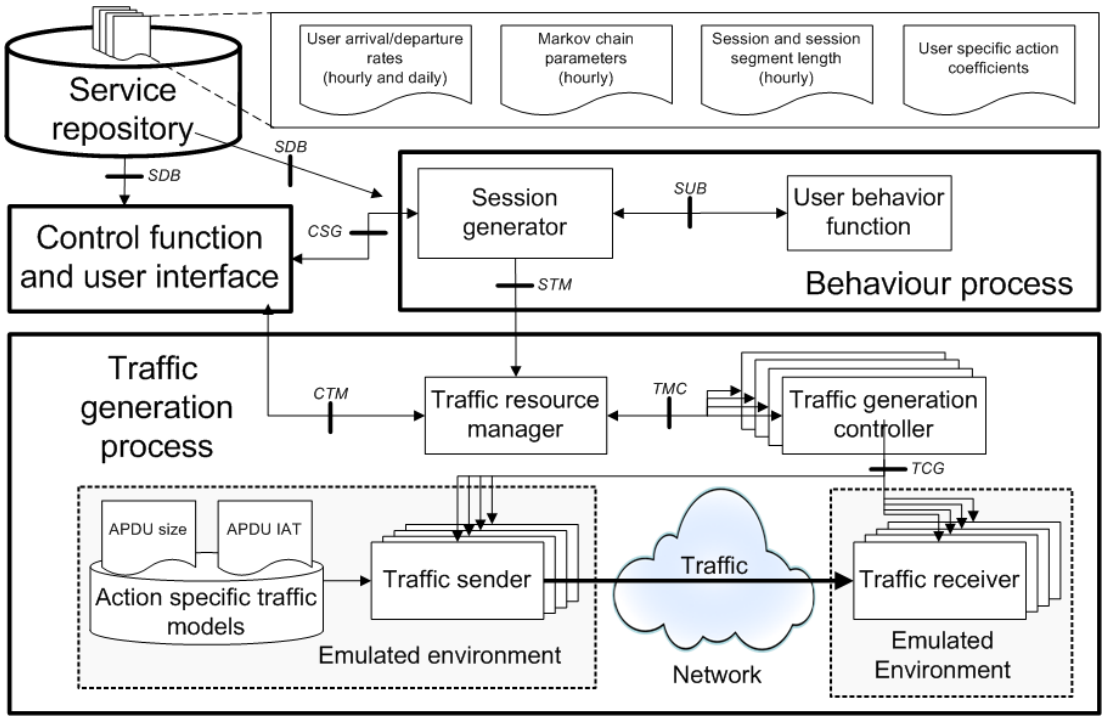

Fig. 1. Functional architecture of UrBBaN-Gen [3]

traffic characteristics [1]. The following behavioural categories are defined for MMORPGs: Trading (i.e., creation and exchange of virtual goods), Questing (i.e., completing tasks given by the Non-Player Characters (NPCs) performed mostly by player alone), Player versus Player (PvP) combat (i.e., battles between groups of players), Dungeons (i.e., combat between a small group of players and NPCs), and Raiding (i.e., combat between a large group of players and NPCs) [1]. In order to generate synthetic network traffic of MMORPGs, different user behaviours must be taken into account as the bandwidth consumption between some behaviours differs by an order of magnitude.

The most popular MMORPG – World of Warcraft was used to create the corresponding player behaviour and traffic models (for both client-to-server and server-to-client traffic). While MMORPGs are a real time based service depending on timely delivery of information from servers to the clients and vice versa, many of them (including WoW) use TCP. WoW TCP streams are very *thin*, especially client to server traffic with average payload size of 36 bytes (less than a 40 bytes TCP/IP header) [1]. The packet rate fluctuates between 4 and 10 packets per second (pps) in one direction (including ACK packets which carry no payload). The problem with such thin TCP streams is that certain algorithms of the TCP can perform poorly or even degrade Quality of Experience (QoE) of the users (e.g., Nagle algorithm, Delayed Acknowledgment). Also, as there are a lot of such streams coming and going from game servers, congestion avoidance algorithms are not very effective. Therefore, to study the effects of MMORPG traffic in realistic conditions, there is a need for a tool like UrBBaN-Gen which can generate not just high packet loads, but a large number of TCP streams as well.

B. UrBBaN-Gen architecture

Figure 1 shows the functional architecture of UrBBaN-Gen, consisting of four modules: *Service repository*, *Control function and user interface*, *Behaviour process*, and *Traffic generation process*. Briefly, the modules correspond to the following functionality:

- the *Service repository* contains the input data which fully describe the simulated service in terms of mathematical models and statistics;
- the *Control function and user interface* are used for managing the parameters of the simulation;
- the *Behaviour process* is in charge of simulating the user behaviour and generating sessions; and,
- the *Traffic generation process* transforms the player behaviour to network traffic.

As already mentioned before, the focus of this paper is on the *Traffic generation process*. The *Traffic generation process* encompasses virtualization and packet flow generation. In the initial implementation of UrBBan-Gen, we used virtualization based on Linux containers to generate a large number of thin TCP streams, by having one instance of a *Traffic sender* or a *Traffic receiver* run within a container. While this implementation enabled us to generate a fairly large number of TCP streams, it has also proved to be not very stable. Several security procedures had to be implemented in order to ensure proper initiation of the Linux containers, proper connection establishing from the senders and the receivers within the containers, and also crash recovery. These faults (especially of the Linux containers as a rather new technology) were some of the main motives for moving on to IMUNES. An additional benefit of using IMUNES is that the emulated environment can also contain other network nodes, as well as

entire IP subnets, which enables running complex experiments in an fully emulated testbed (thus saving a possibly significant amount of time and money for creating a real network testbed).

For the functional elements of *Traffic sender* and *Traffic receiver* we used the Distributed Internet Traffic Generator (D-ITG), an open source tool developed at Universita' degli Studi di Napoli "Federico II" (Italy). D-ITG is a tool for network traffic generation which offers a choice of various transport and application layer protocols.

A very important feature of D-ITG is easy implementation of new application protocols. Newly implemented application protocols are defined by the underlying transport protocol, distribution of the Application Protocol Data Unit (APDU) size and distribution of the inter-arrival time (IAT) of the APDUs (i.e., time passed between sending two subsequent APDUs). APDUs can be "fit" into one packet, or split between several packets depending on the APDU size and the maximum transmission unit (MTU) of the underlying network.

D-ITG's distributed architecture includes sender, receiver, logger, and manager components. More details regarding D-ITG can be found in the respective publications [23], [24], [25].

IV. Improvements in emulated environment

In this paper we focus on the *Emulated environment* of the traffic generation process as shown in Figure 1. This component has been proven to be the bottleneck when the number of senders/receivers has been increased over 100 [27]. We replace the existing LXC containers implementation with FreeBSD *jails*, a kernel based virtualization system used in IMUNES (Integrated Multiprotocol Network Emulator/Simulator). Also the network configuration is done in an optimized environment (FreeBSD kernel) by using netgraph kernel modules.

A. IMUNES fundamentals

IMUNES is a lightweight kernel level network emulator [5][6]. Four main tools inside the standard FreeBSD kernel are used to provide the emulating environment:

1) FreeBSD jails – a lightweight virtualization solution that enables different containers to share system resources with minimum overhead. It is based on separating system resources as a means of providing a higher level of security without affecting system performance [26]. The main advantage of jails is that they run on the same kernel and enable full binary compatibility with FreeBSD executables. All FreeBSD and most Linux applications can run without recompilation. If recompiling is needed the changes to the original source are minimal. Each jail has its own:
   - directory subtree – root file system,
   - hostname,
   - IP address – crucial for achieving network emulation and communication between emulated nodes.
2) Clonable network stack – A jail has a complete instance of the network stack. This is enabled by the clonable network stack that is described in [6].

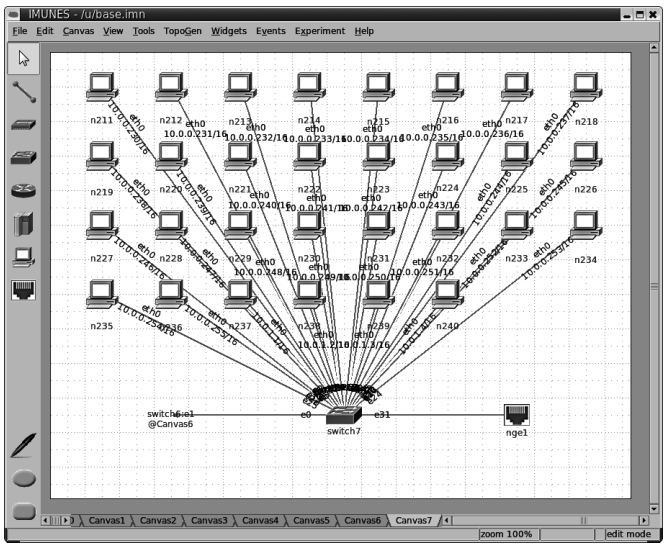

Fig. 2. IMUNES GUI: View of the testbed canvas

3) Netgraph kernel modules – used for emulating node network interfaces and linking virtual nodes in the simulation. Netgraph also provides the implementation of lower layer (data link layer) network equipment such as hubs and switches [28].
4) ZFS file system – Transactional file system originally developed by Sun Microsystems. ZFS uses the concept of storage pools to manage physical storage. ZFS has the ability to create snapshots, read-only copies of the file system state. Snapshots can be cloned and replicated. This makes them suitable for creating initial copies of root file systems used by virtual nodes during simulation [29].

The main advantage of IMUNES is the low system footprint that topologies generate. Traffic manipulation is also done efficiently in the FreeBSD kernel, where packets are transferred by passing references rather than copied as they cross the emulated network environment. The system architecture enables fast experiment instantiating and termination. IMUNES is suitable for executing multiple experiments at once, and its GUI also facilitates creation of large topologies with multiple canvas support (Figure 2) [30]. Large topologies can be divided into canvases that simplifies their management.

B. IMUNES nodes

The IMUNES system has a set of nodes that can be instantiated and configured when a simulation is started. Nodes can be seen in the left sidebar inside the IMUNES GUI [30] (Figure 2). These nodes can be grouped according to the usual reference model layers into three groups: the Layer 1 or Physical layer nodes, the Layer 1/2 nodes, and the Layer 3 nodes.

Their roles may be briefly described as follows:

- Physical layer nodes are used for interconnecting network nodes and creating the topology. The first node in this category are links that create all the paths between

network nodes. Links can be configured to emulate bandwidth, delay, bit-error-rate (BER) and duplicate packets. There is also the physical interface node that enables the connection of IMUNES nodes to the external network. By using this node IMUNES can route real-world traffic and manipulate traffic with link settings. Both physical nodes are part of the netgraph suite (ng_pipe, ng_ether).

- Layer 1/2 nodes emulate hub and layer 2 switch nodes. Both are implemented as netgraph nodes (ng_hub, ng_bridge) and used for creating more complex local area connections. The main difference is that the hub node, when it receives a packet, forwards it to all the other interfaces, whereas the switch node forwards packets based on link layer data.
- Layer 3 nodes operate at the IP layer and up. These nodes are in fact jails with their own set of processes and file systems. IMUNES allows us to create three types of jailed nodes:
  - PC: An empty jail that is, by default, created without any processes running. The PC is an example of a persistent jail [26]. This node is a base for creating all other layer 3 nodes.
  - Host: A jail used for running services. By default it has the *inetd* and *rpcbind* processes running.
  - Router: A node for emulating real routers. It can run the Quagga routing protocol suite [31], or simply be a static router that needs to be manually configured. The Quagga routing suite offers support for most widely used routing protocols: RIP, RIPng, OSPFv2, OSPFv3, BGP, etc. IMUNES is setup to automatically configure RIP(ng) and OSPFv2/3. If needed, other routing protocols and suites can be manually configured to run on the router node.

C. IMUNES batch mode

While the IMUNES GUI provides the most convenient means to create and customize arbitrary network topologies, IMUNES experiments can also be run and managed without the GUI. Experiments can be started and stopped from a terminal shell. To start an IMUNES topology from a shell, the following command needs to be issued:

```
%sudo imunes -b topology.imn
Creating node n0
...
Creating link l0
...
Configuring node n0
...
Configuring node n8
Network topology instantiated in 2 seconds.
Experiment ID = i369b0
```

Once the experiment is started, it can be further managed through the GUI, or through a command line interface. This experiment can be easily shutdown with the use of the same command and providing the Experiment ID (i.e. `i369b0`):

```
%sudo imunes -b -e i369b0
Terminating processes in vimage n0
...
Shutting down vimage n0
...
Shutting down vimage n8
Cleanup completed in 8 seconds.
```

During the experiment, one can also manage virtual links and nodes in IMUNES by using the following scripts available within the standard IMUNES distribution:

- `himage` is used for executing arbitrary commands in a virtual node. Can be used for starting and terminating applications, collecting data from the virtual node and configuring the virtual node.
- `hcp` is used for copying data to the virtual node, and from the virtual node, i.e. copying configuration files, extracting server log files.
- `vlink` is used for changing the properties of the links in network topology within the experiment. The link settings which can be changed include bit-error-rate, bandwidth, delay, and packet duplication rate.

D. Using D-ITG in IMUNES

The main problem encountered when attempting to replace LXCs with IMUNES was the lack of native support for BSD operating systems in D-ITG. (D-ITG has primarily been designed for Windows and Linux operating systems.) As of late 2012, the latest available D-ITG (version 2.8.0-rc1) needed some additional work to recompile successfully on FreeBSD. Once that was completed, the process of integration with IMUNES was fairly straightforward. The D-ITG executables have been inserted into the ZFS snapshot which is replicated across nodes in the IMUNES simulation. In this way, each node in a simulation has gained access to D-ITG binaries.

V. Scalability testing methodology

Two testbeds were created: 1) Linux (LXC) testbed, and 2) IMUNES testbed. The testing was performed on commodity hardware PC-s (Intel Core i3-2120 3.3 Ghz with 4GB of RAM). The same hardware configuration was used for both testbeds.

Linux testbed architecture is illustrated in Figure 3. It consisted of two PCs running Ubuntu 11.10 operating system. One PC hosted all LXCs hosting D-ITG senders connected to a Linux bridge, while other PC hosted D-ITG receivers. The number of instances of LXCs varied depending on the experiment. The IMUNES testbed was also set on 2 PCs, one for running the IMUNES system with all D-ITG sender nodes in an emulated environment and the other that acted as a receiver. The IMUNES system is running on top of FreeBSD 8.3-RELEASE. The D-ITG receiver node was run on another PC so that it would not interfere with the testbed and testing results. The IMUNES testbed was a classic IMUNES experiment that consisted of 240 nodes divided into 8 similar canvases, with 30 nodes per canvas.

The following three tests were run in both testbeds:

1) *Test 1*: Generating UDP packet flows with fixed pps rate and fixed packet size, while increasing the number of sender nodes.

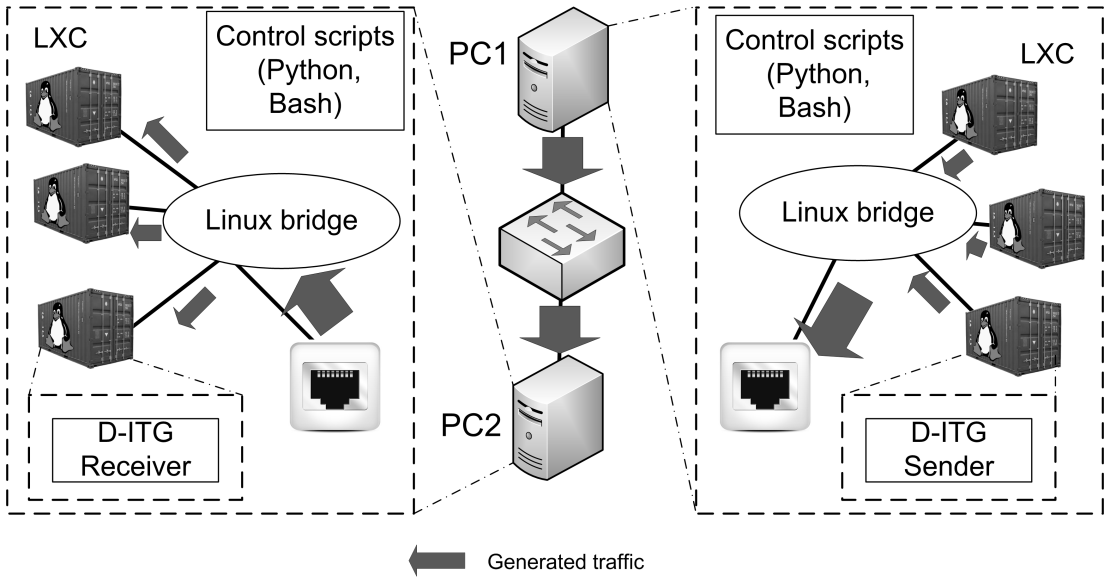

Fig. 3. Linux testbed architecture

2) *Test 2*: Generating UDP packet flows with fixed pps rate and fixed number of nodes, while changing the packet size.
3) *Test 3*: Generating TCP flows with fixed numbers of senders and receivers, while changing the number of generated flows per each sender. This test uses client to server traffic model of *Dungeons* behavioural category of WoW.

For each test iteration the CPU load and packet loss were noted.

Additionally, *Test 4* was run only by using the IMUNES testbed on top of VMware. Through this test we inspected how the UDP packet rate created from a single sender to a single receiver varies in the network emulated by IMUNES.

VI. Results

The IMUNES testbed has shown to be, in overall, more stable than the Linux testbed. The reason is that jails started inside IMUNES is more lightweight and also represents a more "mature technology" in comparison to the Linux containers used in the Linux testbed. Also, results regarding CPU use and packet loss in first two (UDP) tests show that IMUNES is a significantly better solution for high packet loads, and even in the third (TCP) test, IMUNES shows slightly better results.

A. Test 1 results

In this test we investigated the impact of increasing the number of sender nodes to CPU load and packet loss values. The test was done as follows. The packet rate was fixed at 1000 pps and the packet size was 64 bytes. The number of nodes was gradually increased from 10 nodes to 180 nodes.

The results regarding the loss depending of the number of nodes can be seen in Figure 4. It can be observed that after increasing the number of nodes over 40 (i.e., increasing packet rate over 40,000 pps) loss values of Linux testbed increase rapidly. IMUNES had a much smaller packet loss

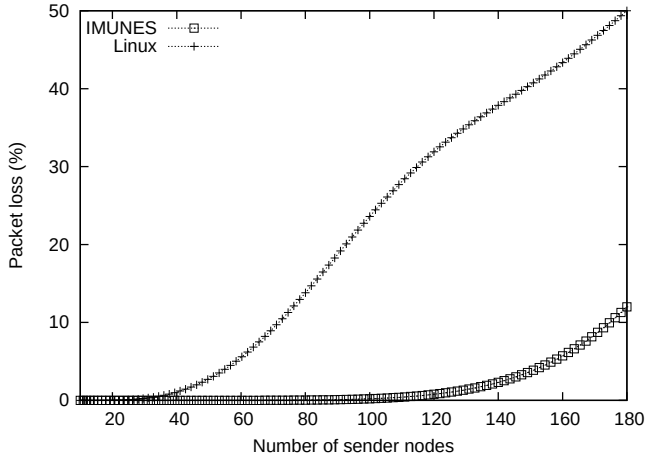

Fig. 4. Packet loss with respect to the number of sender nodes

which started to occur at around 120 nodes (i.e., packet rate of 120,000 pps).

In terms of CPU load, the IMUNES testbed had a linear growth compared to the Linux testbed, which grew faster up to 60 nodes (Figure 5). This behaviour can be explained by the fact that packet loss values of Linux testbed at 60 nodes were very significant, while the IMUNES testbed was more stable because the packet loss was substantially smaller.

B. Test 2 results

The second test used a fixed number of 100 nodes and the packet rate of 200 pps. The packet size was changed from 64 bytes to 1472 bytes because the MTU size was set to 1500 bytes (IP header is 20 bytes and the UDP header is 8 bytes). The results can be seen in Figure 6, depicting the CPU load depending on the size of the packets generated. As it can be seen, the Linux implementation varies significantly in load while the IMUNES is much more stable. This is

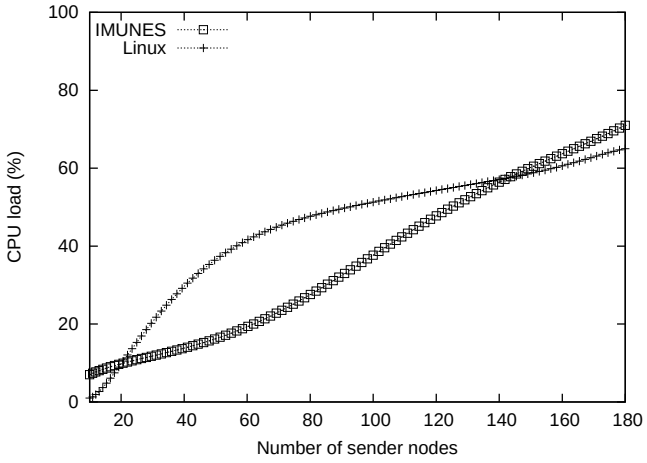

Fig. 5. CPU load with respect to the number of sender nodes

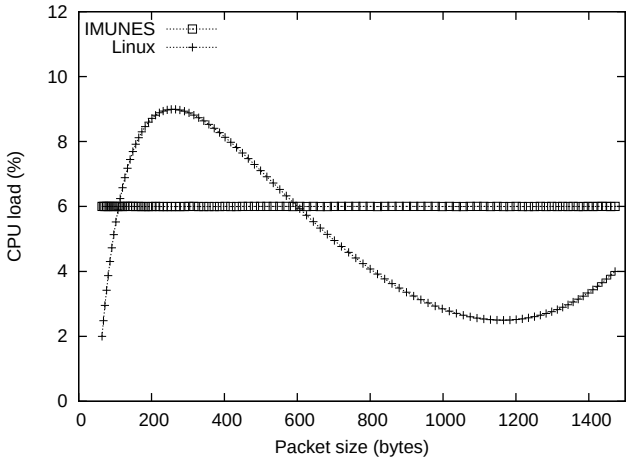

Fig. 6. CPU load with respect to packet size

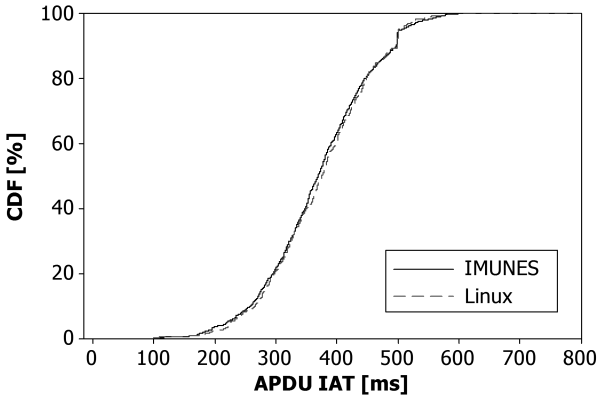

Fig. 7. CDF of APDU IAT with 10,000 active flows

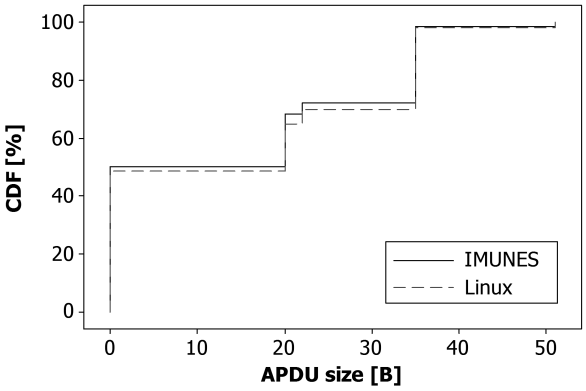

Fig. 8. CDF of APDU size with 10,000 active flows

probably due to the fact that IMUNES handles packets more efficiently. Since the CPU load is rather low it is probably directly connected to the base testbed load caused by the operating systems running on testbeds.

C. Test 3 results

During the third set of tests which used gaming traffic models for generation of TCP traffic we encountered several problems.

The first problem was that D-ITG receiver in version 2.8 would randomly start to decrease the TCP sending window size. We believe that this is due to the receiver not being able to process those packets. Once the sending window size of the receiver reached zero, the receiver would send a message with "Zero Window Size" to the sender which would stop generation of the packets. This error was not present in the previously used D-ITG version 2.7. We established that such behaviour is only seen when complex distributions of the APDU size (e.g., Weibull distribution) were used in the model of the traffic. As we could not find the solution for this error, we remodelled the client to server APDU size of the Dungeons category using Normal distribution, and used that for testing purposes. We aim to investigate further this issue and find a solution in future work.

Additionally, we encountered an operating system boundary on the number of created TCP flows in both testbeds. We successfully created more than 10,000 TCP flows but the traffic was inconsistent. At 10,000 flows both systems had similar CPU load at around 20%. There was no packet loss on the TCP flows as each generated TCP flow had on average 2.3 packets per second (in one way) which is consistent with the UDP results where loss occurs at pps rates of 40,000 pps and up.

In final setup, the third test had a fixed number of 240 sender nodes and the traffic was generated according to the "Dungeons" client to server traffic model. The model was adjusted to use the normal distribution (not Weibull) due to the "Zero Window Size" error. We gradually increased the number of active flows and monitored CPU load, packet loss, and also the statistical characteristics of the generated traffic to make sure that the properties of the traffic conform to the defined statistical models.

Figure 7 and Figure 8 show the characteristics of one flow captured randomly from 10,000 active flows. Both figures show that distributions of APDU size and inter arrival times (IAT) do not deteriorate when the load is increasing for both Linux and IMUNES testbed. We can conclude that the

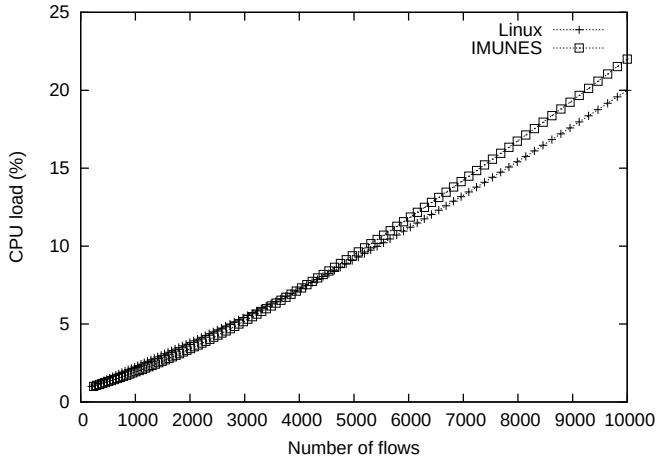

Fig. 9. CPU load with respect to number of TCP flows

IMUNES testbed does not deteriorate the statistic properties of the game traffic in comparison with the Linux testbed which was thoroughly tested in terms of statistical accuracy in previous work [3].

Figure 9 demonstrates how both testbeds deal with the number of flows in a similar way, while keeping the CPU load fairly low, (at below 25are also consistent with the traffic characteristics. The CPU load values and increase rate suggest that the system could manage more than 10,000 TCP flows after tuning the operating system. This will be explored in future work.

D. Test 4 results

As previously stated, the last test was run only on IMUNES on top of VMware. Figure 10 shows the maximum achieved packet rate with zero loss for each packet size. We used the packet size of 64, 512, and 1500 bytes for 1, 10, 50, 100, and 250 streams. It can be noted that the highest packet rate without loss has been achieved for the packet size of 64 bytes.

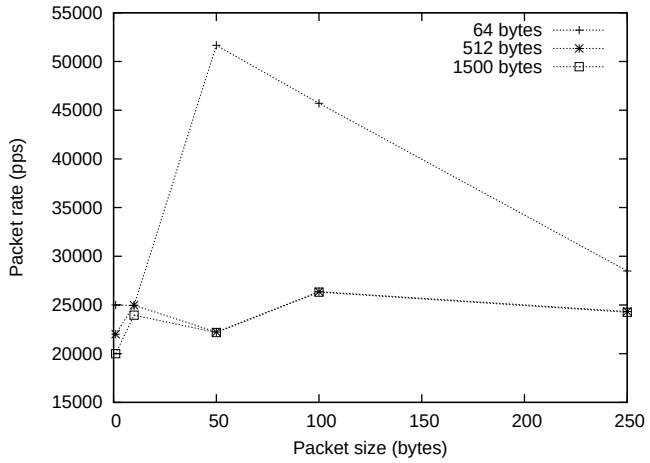

Fig. 10. Maximum achievable zero-loss packet rate with respect to packet size (IMUNES)

VII. Conclusion

In this paper we have presented improvements to our architecture for user behaviour based traffic generation. Linux containers, the previously used technology for virtualization has been replaced with IMUNES. Both testbeds have been tested in order to compare CPU load and packet loss while generating traffic at higher packets per second rates. The implementation with IMUNES showed better results in both terms of lost packets and CPU load. Performance evaluation was done with high packet rates and with a high number of gaming flows. Shown results enable generation of MMORPG traffic on a truly massive scale. We achieved generating 10,000 MMORPG gamers on a single PC. For our future work we aim to add expandability into the IMUNES prototype so that multiple PCs can participate in a single simulation.

Acknowledgments

This work was supported by the research projects 036-0362027-1639 and 036-0362027-1640, funded by the Ministry of Science, Education, and Sports of the Republic of Croatia and the E-IMUNES project funded by Ericsson Nikola Tesla, Zagreb, Croatia. Also, the research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no. 285939 (ACROSS).

References

[1] M. Suznjevic, O. Dobrijevic, and M. Matijasevic, "MMORPG player actions: Network performance, session patterns and latency requirements analysis," *Multimedia Tools and Applications*, vol. 45, no. 1-3, pp. 191–241, 2009.

[2] Cisco Systems, "Cisco Visual Networking Index: Forecast and methodology, 2010-2015," 2011.

[3] M. Suznjevic, I. Stupar, and M. Matijasevic, "A model and software architecture for MMORPG traffic generation based on player behavior," *Multimedia Systems*, DOI: 10.1007/s00530-012-0269-x, 2012.

[4] V. Vasic, M. Suznjevic, M. Mikuc, and M. Matijasevic, "Improving distributed traffic generation performance by using IMUNES network emulator," in *Proceedings of the 2012 International Conference on Software, Telecommunications and Computer Networks*, p. 5, 2012.

[5] M. Zec and M. Mikuc, "Real-time network IP network simulation at gigabit data rates," in *Proceedings ConTEL 2003*, pp. 235-242, 2003.

[6] M. Zec and M. Mikuc, "Operating system support for integrated network emulation in IMUNES," in *1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS)*, pp. 3-12, 2004.

[7] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of the ACM SIGMETRICS*, pp. 151–160, 1998.

[8] P. E. Heegaard, "Gensyn - a Java based generator of synthetic Internet traffic linking user behaviour models to real network protocols," in *ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, 2000.

[9] P. Svoboda, W. Karner, and M. Rupp, "Traffic analysis and modeling for World of Warcraft," in *IEEE International Conference on Communications, 2007. ICC '07.* , pp. 1612–1617, 2007.

[10] J. Kim, E. Hong, and J. Choi, "Measurement and Analysis of a Massively Multiplayer Online Role Playing Game Traffic," in *Proceedings of Advanced Network Conference*, pp. 1–8, 2003.

[11] H. Park, T. Kim, and S. Kim, "Network traffic analysis and modeling for games," in *Internet and Network Economics*, Lecture Notes in Computer Science, pp. 1056–1065, Springer Berlin / Heidelberg, 2005.

[12] M. Suznjevic and M. Matijasevic, "Player Behavior and Traffic Characterization for MMORPGs: A Survey," *Multimedia Systems*, DOI: 10.1007/s00530-012-0270-4, 2012.

[13] X. Wang, T. Kwon, Y. Choi, M. Chen, and Y. Zhang, "Characterizing the gaming traffic of World of Warcraft: From game scenarios to network access technologies," *IEEE Network*, vol. 26, no. 1, pp. 27-34, 2012.
[14] J. Saldana, M. Suznjevic, L. Sequeira, J. Fernandez-Navajas, M. Matijasevic, J. Ruiz-Mas, "The Effect of TCP Variants on the Coexistence of MMORPG and Best-Effort Traffics," in *Proceedings of the 21st International Conference on Computer Communications and Networks (ICCCN)*, p.5, 2012
[15] K. Shin, J. Kim, K. Sohn, C. J. Park, and S. Choi, "Transformation Approach to Model Online Gaming Traffic," *ETRI Journal*, vol. 33, no. 2, pp. 219–229, 2011.
[16] K. Shin, J. Kim, K. Sohn, C. Park, and S. Choi, "Online gaming traffic generator for reproducing gamer behavior," in *Proceedings of the 9th international conference on Entertainment computing*, pp. 160–170, 2010.
[17] D. Drajic, S. Krco, I. Tomic, P. Svoboda, M. Popovic, N. Nikaein, and N. Zeljkovic, "Traffic generation application for simulating online games and M2M applications via wireless networks," in *Proc. of the 2012 9th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 167 – 174, 2012
[18] M. Zec, L. Rizzo, and M. Mikuc, "DXR: towards a billion routing lookups per second in software," *Computer Communication Review*, DOI: http://dx.doi.org/10.1145/2378956.2378961, 2012
[19] J. Arenholz, C. Danilov, T.R. Henderson, and J.H. Kim, "CORE: A real-time network emulator," in *IEEE MILCOM*, 2008.
[20] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.
[21] M. Suznjevic, I. Stupar, and M. Matijasevic, "MMORPG player behavior model based on player action categories," in *Proceedings of the 10th Workshop on Network and System Support for Games*, p. 6, 2011.
[22] M. Suznjevic, I. Stupar, and M. Matijasevic, "Traffic modeling of player action categories in a MMORPG," in *Proceedings of the 2nd workshop on DIstributed SImulation and Online gaming (DISIO)*, p. 8, 2011.
[23] S. Avallone, S. Guadagno, D. Emma, and A. Pescapé, and G. Venturi, "D-ITG Distributed Internet Traffic Generator," in *Proceeding of International Conference on Quantitative Evaluation of Systems*, pp. 316–317, 2004.
[24] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 158 – 165, 2004.
[25] A. Botta, A. Dainotti, and A. Pescapè, "Multi-protocol and Multi-platform Traffic Generation and Measurement," in *INFOCOM 2007, 26th IEEE International Conference on Computer Communications, Demonstration Session*, 2007.
[26] P.H. Kamp and R.N.M. Watson, "Jails: Confining the omnipotent root," in *2nd International SANE Conference*, p. 15, 2000.
[27] M. Suznjevic, "*Modelling of network traffic for multiplayer role playing games based on user behaviour*," PhD Thesis, University of Zagreb, Croatia, May 2012.
[28] A. Cobbs, "All about nethgraph." http://people.freebsd.org/ julian/netgraph.html.
[29] J. Bonwick and B. Moore, "ZFS: The last word in filesystems." http://hub.opensolaris.org/bin/view/Community+Group+zfs/WebHome.
[30] M. Zec, M. Mikuc, A. Mijocevic, S. Marjanovic, and V. Vasic, "Imunes manual." http://imunes.tel.fer.hr/imunes/dl/imunes$_ug2$0110907.*pdf*.
[31] "Quagga project, quagga routing suite." http://www.quagga.net.

Valter Vasić received his M.Sc. in Computer Science in 2010 from the Faculty of Electrical Engineering and Computing, University of Zagreb. He is currently employed as a research assistant at the same faculty within the E-IMUNES project funded by Ericsson Nikola Tesla. His research interests include security, network simulation and virtualization. He is an author of 5 conference papers and a member of IEEE.

Mirko Sužnjević received his PhD in Electrical Engineering in 2012 from the Faculty of Electrical Engineering and Computing, University of Zagreb. He is currently employed as a senior researcher at the same faculty within the ACROSS Project funded by the Seventh Framework Programme of the European Union. His research interests include analysis and modelling of network traffic, cloud computing, and Quality of Service/Experience. His research is mostly focused on the complex multimedia services and especially online games. He is an author of 4 journal and 9 conference papers. He is a member of IEEE and an active contributor in the IETF.

Miljenko Mikuc Miljenko Mikuc received his PhD in Electrical Engineering from University of Zagreb, Croatia, in 1997. He is currently Associate Professor at the Faculty of Electrical Engineering and Computing, Department of Telecommunications within the same university. His area of interest includes network protocols, network simulation and security.

Maja Matijašević is a Professor in the Faculty of Electrical Engineering and Computing at the University of Zagreb (FER), Croatia, and the leader of the Networked Media research group within the FER's Department of Telecommunications. Her research interests include networked multimedia and quality of service in IP-based next generation networks, with particular focus on session negotiation, adaptation, and mobility. She is a principal researcher in a Croatian national research project and a research program, and she has led research projects in collaboration with industry. She has over 80 journal and conference publications, and she has coauthored several books and book chapters. She has served as a TPC member, TPC (co)chair, and reviewer in international conferences, and as a guest editor in several journal special issues. She received her Dipl.-Ing, M.Sc. and Ph.D. degrees in Electrical Engineering from the University of Zagreb and the M.Sc. in Computer Engineering from the University of Louisiana at Lafayette, LA, USA. She is a Senior Member of IEEE and a member of ACM.