Improving distributed traffic generation performance by using IMUNES network emulator

Valter Vasić, Mirko Sužnjević, Miljenko Mikuc, Maja Matijašević
University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia
{valter.vasic, mirko.suznjevic, miljenko.mikuc, maja.matijasevic}@fer.hr

***Abstract*—In this paper we present improvements of our software architecture for User Behaviour Based Network Traffic Generation (UrBBan-Gen). It consists of four modules *Service repository*, *Control function and user interface*, *Behaviour process*, and *Traffic generation process*. Improvements are performed on the traffic generation process, specifically on the virtualization section of UrBBaN-Gen through integration of the Distributed Internet Traffic Generator (D-ITG) which we use as a module for traffic generation with network simulator and emulator IMUNES. In this way we achieve two goals: 1) better scalability of the traffic generation and 2) possibility for testing of various network scenarios in simulated networks under realistic loads.**

1. Introduction

In recent years Massively Multiplayer Online Role-Playing Games (MMORPGs) have become a growing phenomenon which attracts millions of players. 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 [1], gaming traffic will grow with a compound annual growth rate of 43% in a period of 2010–2015, a second largest growth after the video category. These games generate traffic that is very demanding in terms of Quality of Service (QoS) due to the real-time nature of virtual worlds of networked games.

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 providing connectivity for these games needs to be well designed and tested. Generating realistic traffic loads is one of the most important tasks in network testing.

In our previous work [2] 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 generate the traffic of hundreds and thousands of players. To achieve this we employed two scalability techniques: 1) virtualization and 2) expandability (additional PCs can simply be added into the traffic generation process).

In this paper we further improve UrBBan-Gen in the aspect of scalability. We replace previous virtualization technology (Linux Containers - LXC) through integration of UrBBan-Gen with network simulator and emulator IMUNES [3][4]. We demonstrate the gains through several measurements focusing on generated packed load and bandwidth load, and measuring saturation of the CPU and RAM usage of the PC participating in the traffic generation. This integration also enables testing 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.

The remainder of the paper is organized as follows: in section 2 we present the related work in the area of behaviour based traffic generation with emphasis on networked games and virtualization techniques, in section 3 we describe the UrBBaN-Gen, in section 4 we explain improvements of the emulated environment, section 5 contains the testing methodology applied, section 6 the results of the measurements, and we conclude the paper in section 7.

2. Related work

Network traffic generators (TGs) may be categorized based on how they derive the traffic patterns into three groups: 1) TGs based on replay of previously captured traffic, 2) TGs emulating the properties of previously captured traffic based on statistical analysis of the traces, and 3) source based TGs, 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) [5]. 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.

GenSyn is a synthetic traffic generator implemented in Java based on user behaviour [6]. 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.

While there are several models of network traffic of MMORPGs described in literature [7], [8], [9] there are very

few implementations of models in traffic generators. Shin et al. [10] propose a novel method for modelling the network traffic of games. They analyse packet size and inter-arrival times of *World of Warcraft* (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 a online game traffic generator [11]. 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 [2] is fully defined by application level user behaviour.

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 [3], [4]. IMUNES runs natively on FreeBSD, and it can also run in virtual machine software such as VMware. The CORE network emulator was created as an offspring of IMUNES [12]. 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 [13] 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.

3. URBBAN-GEN

The goal of UrBBaN-Gen is to generate realistic traffic of a MMORPG, as a complex IP service, on both client and server side and based on user behaviour. This is achieved through implementing a source based traffic model based on player behaviour on the application level defined through action categories [14] and traffic models of each for each action category [15]. While UrBBaN-Gen has been developed to generate WoW network traffic, functional architecture and implementation are service independent so new services may be added through new user behaviour and traffic models. MMORPGs are a good case study as they involve large number of users with diverse application level behaviours which significantly affect network traffic characteristics [16].

Figure 1 shows the traffic generation part of UrBBaN-Gen. The higher layers perform simulation of the player behaviour and control the traffic generation process.

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. D-ITGs distributed architecture includes sender, receiver, logger, and manager components. More details regarding D-ITG can be found in several publications of its authors [17], [18], [19].

4. IMPROVEMENTS OF 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. We replace the existing LXC containers implementation with FreeBSD jails, a kernel based virtualization system used in IMUNES. Also the network configuration is done in an optimized environment (FreeBSD kernel) by using netgraph kernel modules.

4.1 IMUNES fundamentals

IMUNES (Integrated Multiprotocol Network Emulator/Simulator) is a lightweight kernel level network emulator [3][4]. Three main tools inside the standard FreeBSD kernel are used to provide the emulating environment:

1) FreeBSD jails - a lightweight virtualization solution that enables different jails 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 [20]. The main advantage of jails is that they run on the same kernel and enable full binary compatibility with FreeBSD executables. Most Unix and Linux applications can be 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. A jail has a complete instance of the network stack. This is enabled by the clonable network stack that is described in [4].
2) 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 [21].
3) ZFS file system - Transactional file system 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 [22].

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 instantiation and termination.

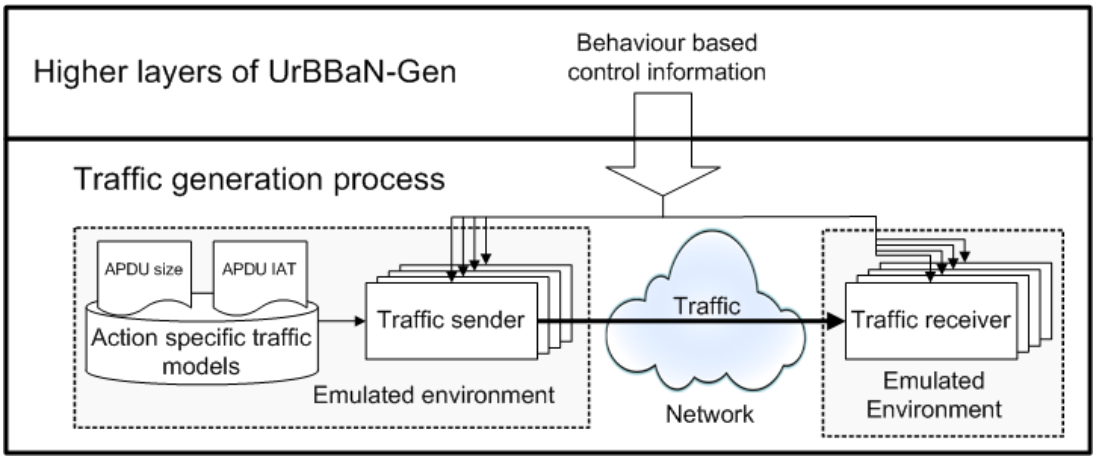

Fig. 1. Architecture of UrBBaN-Gen functionality [2]

IMUNES is suitable for executing multiple experiments at once, and it also facilitates creation of large topologies with canvas support [23].

4.2 IMUNES nodes

The IMUNES system has a set of nodes that can be instantiated and preconfigured when a simulation is started. Nodes can be seen in the left sidebar inside the IMUNES GUI [23]. These nodes can be grouped in the following groups:

- Physical layer nodes - 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 network problems and properties such as bandwidth, delay, bit-error-rate (BER) and duplicate packets. There is also the physical interface node that enables the connection of all the IMUNES nodes to the external network by assigning a NIC (Network Interface Card) to the physical interface node. 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 implemented as netgraph nodes (ng_hub, ng_bridge). 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 - Include nodes that operate from the IP layer 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 created, by default, without any processes running. The PC is an example of a persistent jail [20]. This node is a base for creating all other layer 3 nodes.
  - Host - Jail used for running services. By default it has the inetd and rpcbind processes running.
  - Router - This node is used for emulating real routers. It can run the Quagga routing protocol suite [24], 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.

4.3 D-ITG in IMUNES

The main issue regarding replacing LXCs with IMUNES is that D-ITG does not have innate support for BSD operating systems (OSs) as it is designed for Windows and Linux OSs. The latest D-ITG version 2.8.0-rc1 would not compile by default on FreeBSD.

After all errors were solved and D-ITG was successfully recompiled on a FreeBSD system the process of integration in IMUNES is fairly simple. The D-ITG executables are inserted into the ZFS system snapshot which is replicated across nodes in the IMUNES simulation. In this way, each node in a simulation has access to D-ITG binaries.

5. 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). For both testbeds the same hardware was used.

Linux testbed 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 D-ITG receiver node was run on another PC so that it would not interfere with the testbed and testing results.

Two tests were performed on both testbeds:

1) Testing with fixed pps (packets per second) rate and fixed packet size, while changing the number of sender nodes.
2) Testing with fixed pps rate and fixed number of nodes, while changing the packet size.

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

The third test was run only on the IMUNES testbed on top of VMware. Through this test we inspected how the packet rate created by a single sender to a single receiver varies in the IMUNES emulated network.

6. Results

The IMUNES testbed was, in overall, more stable than the Linux testbed. The reason is due to jails started inside IMUNES were more lightweight in comparison to the Linux containers used for the Linux testbed.

The first test was done as follows. The packet rate was fixed to 1000 pps and the packet size was 64 bytes. The number of nodes was gradually increased from 10 nodes to 180 nodes. IMUNES had a much smaller packet loss while the Linux testbed started losing packets much more quickly. The results can be seen in Figure 2.

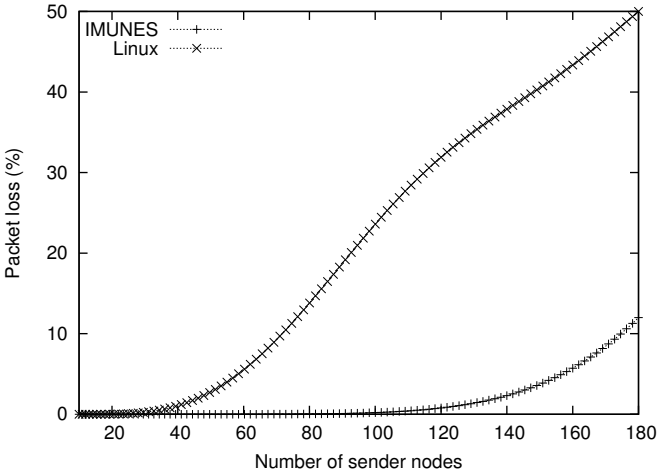

Fig. 2. Packet loss with respect to the number of sender nodes

The IMUNES testbed had much less packet loss than the Linux testbed and the CPU load successfully reached 100%. From 2 we can see that at generation rate of 180 000 packets per second (180 nodes at a rate of 1000pps) IMUNES had only 10% loss while the Linux implementation reached 50%.

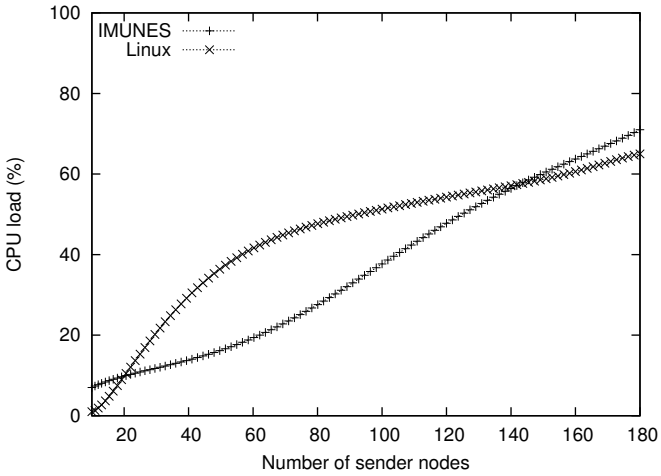

Fig. 3. CPU load with respect to the number of sender nodes

The IMUNES testbed in terms of CPU load had a linear growth compared to the Linux testbed, which grew faster up to until 60 nodes. This can be seen in Figure 3. Obviously the IMUNES tesbed was more stable because the packet loss was substantially smaller than in the Linux testbed.

The second test had 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 4 depicting the load of the processor 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.

As previously stated, the third test was run only on IMUNES which was run on top of VMware. In Figure 5 results of this experiment are depicted so that for each packet size, the maximum achieved packet rate without loss is shown.

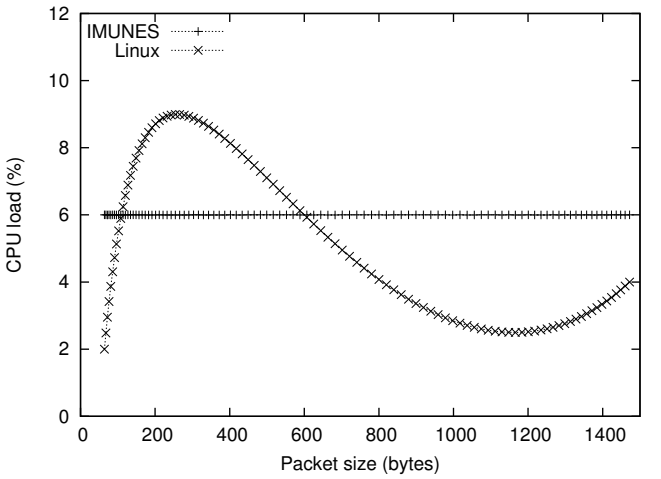

Fig. 4. Processor load with respect to packet size

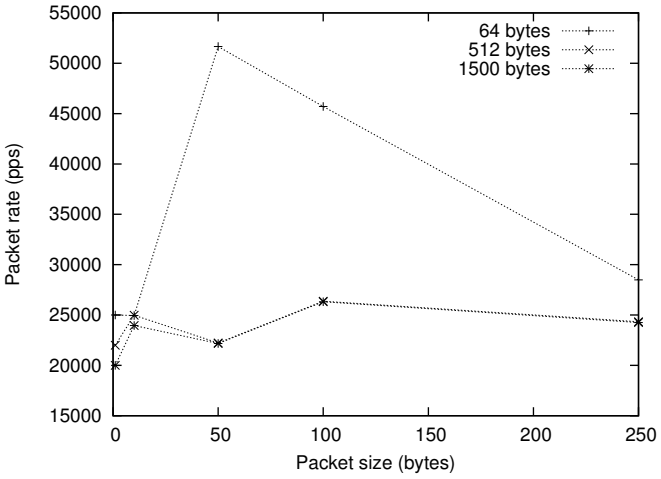

Fig. 5. Packet rate with respect to packet size

We measured for the packet sizes of 64, 512, and 1500 bytes for 1, 10, 50, 100, 250 streams. It can be noted that the highest packet rate without loss has been achieved for the packet size of 64 bytes.

The tests showed that the IMUNES testbed is more stable than the Linux one. Also, results regarding CPU use and packet loss show that IMUNES is a better solution, which additionally offers possibility of emulating complex network scenarios.

7. 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 loads on the processor and packet losses during traffic generation with a high number of packets per second. The implementation with IMUNES showed better results in both terms of lost packets and processor load. Performance evaluation was done with high packet rates. Shown results enable generation of MMORPG traffic on a truly massive scale. For our future work we aim to add expandability into the IMUNES prototype so 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] Cisco Systems, "Cisco Visual Networking Index: Forecast and methodology, 2010-2015," 2011.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] J. Arenholz, C. Danilov, T.R. Henderson and J.H. Kim, "CORE: A real-time network emulator," in *IEEE MILCOM*, 2008.

[13] B. Lantz, B. Heller and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[14] 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.

[15] 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.

[16] 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.

[17] S. Avallone, S. Guadagno, D. Emma, 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.

[18] 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.

[19] 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.

[20] P.H. Kamp and R.N.M. Watson, "Jails: Confining the omnipotent root," in *2nd International SANE Conference*, p. 15, 2000.

[21] A. Cobbs, "All about nethgraph." http://people.freebsd.org/~julian/netgraph.html.

[22] J. Bonwick and B. Moore, "ZFS: The last word in filesystems." http://hub.opensolaris.org/bin/view/Community+Group+zfs/WebHome.

[23] M. Zec, M. Mikuc, A. Mijocevic, S. Marjanovic and V. Vasic, "Imunes manual." http://imunes.tel.fer.hr/imunes/dl/imunes_ug_20110907.pdf.

[24] "Quagga project, quagga routing suite." http://www.quagga.net.