Using NetFPGA to Offload Linux Netfilter Firewall

Mou-Sen Chen[1], Ming-Yi Liao[1], Pang-Wei Tsai[1]
, Mon-Yen Luo[2], Chu-Sing Yang[1*], C. Eugene Yeh[3]
[1]Institute of Computer and Communication Engineering, Dept. of Electrical Engineering,
National Cheng Kung University, Taiwan, R.O.C.
Dept. of Computer Science and Information Engineering,
[2]National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, R.O.C.
[3]National Center for High-Performance Computing, Taiwan, R.O.C.
Email: *csyang@ee.ncku.edu.tw

ABSTRACT
The bandwidth of network traffic has also increased significantly along with the growth of the Internet bandwidth. Network-intensive application systems, such as web server and real-time streaming server, etc, must be capable of filtering malicious packets in a high traffic environment. However, firewall functions and network applications share common CPU resources for server equipping software-based firewall. Moreover, when incoming packets and firewall rules increase, classifying and filtering tremendous attack traffic require significant CPU time and also affect the quality of network applications.
To resolve such problems, this paper proposes a high-speed firewall: NetfilterOffloader firewall implemented in NetFPGA platform, using the NetFPGA to offload the Linux Netfilter firewall and to improve the performance of network applications.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – Security and protection (e.g., firewalls); C.5.5 [**Computer System Implementation**] Servers

General Terms
Measurement, Performance, Design, Experimentation, Security

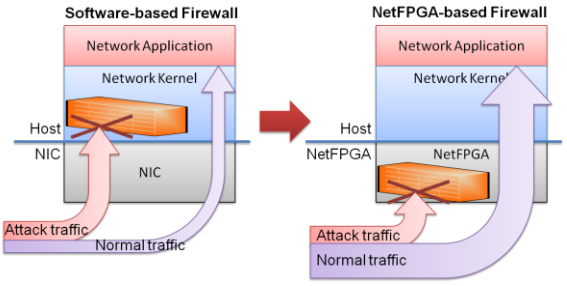
Keywords
Firewall, Netfilter, NetFPGA, Offloading, Prototype

1. INTRODUCTION
The bandwidth of network traffic in edge network has also increased significantly along with the growth of Internet bandwidth and network technology. Many network-intensive application servers (e.g., web servers and real-time streaming servers, etc.) must process tremendous amounts of incoming packets. Besides, malicious traffic, such as viruses and worms, consumes lots of system and network resources to affect the quality of the network application.

Most giant servers build firewalls to filter attack traffic or malicious packets, and many operating systems support the firewall functions. For example, the Linux kernel implements the Netfilter firewall [1]. However, software-based firewalls, such as Netfilter firewall, classify the traffic using general purpose processors; the user-space network application and kernel-space firewall share common CPU resources. Moreover, when tremendous numbers of packets enter the host, in addition to overhead of interrupt handling, the CPU spends considerable time on processing unexpected packets, thereby affecting the network application performance. Since network applications would only be allocated little system resource, the above situations will lead to a reduction in the overall performance of the server.

In order to improve the performance of server built with firewall in high traffic environment, this paper uses the NetFPGA [2] to implement a high-speed firewall, the NetfilterOffloader firewall, to offload the Netfilter firewall function. NetfilterOffloader firewall reduces the Netfilter firewall's loading, and the server can utilize more CPU time for providing more and better services.

The main design concept of this paper is shown in Figure 1. Figure 1(a) shows that the Netfilter firewall blocks an attack packet until packets enter the network kernel. The host spends unnecessary CPU time on filtering attack traffic and causes poor throughput of network application. Figure 1(b) indicates that NetFPGA early filters the attack traffic. The host could reserve more system resources for the network application.



(a) Netfilter firewall **(b) NetfilterOffloader firewall**

Figure 1: Design concept of NetfilterOffloader firewall

The main benefits of NetfilterOffloader are summarized as the following four points:

Load shedding

NetfilterOffloader firewall offloads kernel-space packet filtering function into hardware for reducing the load in the host-end.

High-speed traffic classification

Linux Netfilter firewall performs traffic classification using general purpose processors. NetfilterOffloader firewall on NetFPGA possesses the characteristics of hardware parallel processing and pipeline design. NetfilterOffloader firewall can accelerate the Netfilter firewall to reach high-speed traffic classification.

Early filter/discard

NetfilterOffloader firewall discards unexpected packets early, so as to reserve more resources for normal traffic. The host kernel does not require spending time on processing attack traffic.

Application isolation

NetfilterOffloader firewall prevents unexpected packets from disturbing the processing of normal packets. So, the host kernel does not waste time on undesirable traffic classification and filtering. Furthermore, network applications are isolated from the unexpected traffic in the host kernel. Also, the network application can provide more services to users.

The rest of paper is organized as follows: Section 2 describes the Netfilter framework architecture. Section 3 presents the packet filtering implementation in the NetFPGA platform. Section 4 explains how to combine the hardware firewall with the software firewall. Section 5 shows the performance results. Section 6 describes research works which were similar or related to our work. Section 7 concludes this paper.

2. NETFILTER FRAMEWORK

The Netfilter framework is located in the Linux kernel IP layer; it provides a set of hooks to intercept and manipulate the packets. Netfilter framework provides the packet processing function such as: packet filtering, packet forwarding, connection tracking, Network Address Translation (NAT), and packet mangling for packet modification, etc.

The Netfilter framework for kernel version 2.6 implements five hooks to intercept and manipulate packets as illustrated in Figure 2. If the packets are forwarded to the next hop, they go through the path of PREROUTING, FORWARD, and POSTROUTING chains. The packets are received to local network service via the PREROUTING and INPUT chains. And outgoing packets are sent out via OUTPUT and POSTROUTING chains. Netfilter firewall is registered at INPUT chain for end-host servers.

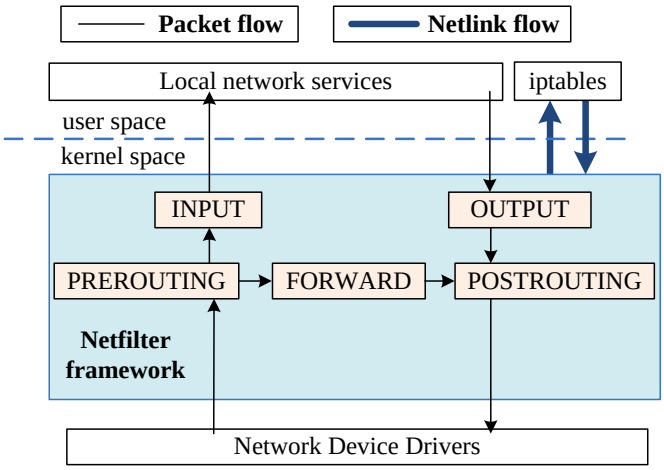


Figure 2: Netfilter framework

Netfilter framework provides the *iptables* utilities for users to configure the Netfilter framework, e.g., firewall rules configuration. The *iptables* utilities in user space communicate with the Netfilter framework in the kernel space via the Netlink socket [3]. Netlink socket is socket-like system calls for accessing the kernel space. Unlike other system calls, Netlink socket has the benefits that support asynchronous operations, duplex characteristics, multicasting and short response time for user-space applications, etc.

Netfilter firewall manages the firewall rules using the linked-list data structure. So every packet must check all firewall rules until it finds the rule-matching result. As a consequence, the number of rules and incoming packets determine firewall's computation complexity. With the growth of rules and incoming packets, CPUs would spend considerable time on the Netfilter firewall; this situation would influence the overall performance of network application.

The next section will describe the packet filtering in NetFPGA. Using NetFPGA to reduce the Netfilter firewall's loading can solve the above mentioned situations, thereby improving the throughput of network application.

3. PACKET FILTERING IN NETFPGA

This section describes the NetfilterOffloader firewall implementation in the NetFPGA platform. The following subsections contain two parts: the first part introduces the NetFPGA platform and implementation on NetFPGA. The second part describes the hardware data path of the NetfilterOffloader firewall.

3.1 NetFPGA Platform

NetFPGA is a high-speed, flexible, and open platform for research; it contains four Gigabit Ethernet interfaces and a Xilinx Virtex-II FPGA programmed with user-defined logic. Stanford University's CS344 course provides open source Verilog designs. Many reference designs were released under open source license, e.g., reference router [4], reference NIC, and OpenFlow switch [5], etc. The above designs were implemented in reusable reference pipeline design [4]. Our packet filtering design is also based on the reference pipeline architecture. Then, the next section describes our hardware data path design.

3.2 Hardware Data Path

The hardware data path of NetfilterOffloader firewall as shown in Figure 3 is based on the reference NIC. The generic *user data path* includes three pipeline modules: *input arbiter*, *output port lookup*, and *output queues* in user data path. The *input arbiter* performs round-robin arbitration to serve one of received queues. *NetfilterOffloader output port lookup* executes packet filtering functions. *Output queues* store the packets in off-chip SRAM or on-chip BRAM until the output port is available.

The block diagrams in *NetfilterOffloader output port lookup* are shown in Figure 3. When packet bus enters *NetfilterOffloader output port lookup*, packets are buffered in *input_fifo* and input into the *packet_extract* module. The *packet_extract* module extracts header information, including 5-tuple fields (source IP, destination IP, source port, destination port, and L4 protocol). After extracting above fields, the *tcam_lookup* module compares 5-tuple fields from packet with the predefined firewall rules in wildcard format. The *tcam_lookup* module was modified from the wildcard lookup module from the NetFPGA OpenFlow switch project [5]; utilized four SRL-based CAMs generated from Xilinx Core Generator, *coregen* utility [6], [7]. While finishing the *tcam_lookup*, the lookup results are pushed into *result_fifo*. The *action_processor* module executes corresponding actions according to the lookup results, e.g., packet drop, forwarding, and slow path to host, etc. However, if no rules are matched or packets come from the host, *action_processor* does the default NIC's jobs that sends packets to corresponding Ethernet ports or CPU ports.

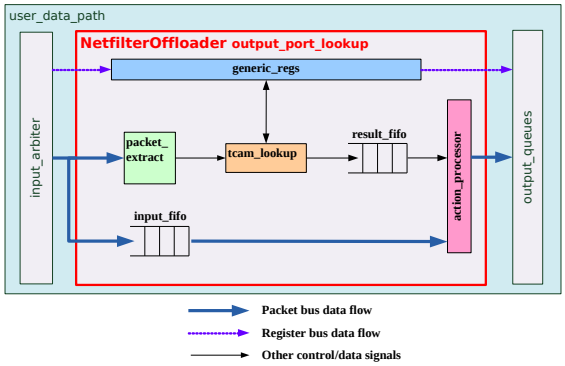


Figure 3: Packet filtering hardware data path

4. SOFTWARE & HARDWARE FIREWALLS INTEGRATION

This section describes how to integrate the NetfilterOffloader firewall with the native Netfilter firewall, including two subsections: First subsection describes the overall software architecture. Second subsection introduces the multi-level traffic classification technique.

4.1 Software Architecture

The overall software architecture is shown as Figure 4. The software components in the host are described as follows.

NF2 driver

The NF2 kernel module was provided from NetFPGA 2.0.0 project. The NF2 kernel module includes the network Gigabit Ethernet interface driver and provides register access using the *ioctl*() kernel interface via PCI bus.

NetfilterOffloader module (NFO module)

The NetfilterOffloader (NFO) module was implemented in loadable kernel module over the NF2 driver. NFO module uses the linked-list data structure for management of TCAM entries in NetFPGA, and provides the Netlink socket kernel interface to replace the *ioctl*(). Since Netlink socket has better response time than *ioctl*() system calls.

Iptables

Iptables was patched to support both Netfilter and NetfilterOffloader firewalls using the Netlink socket system call.

Netfilter firewall

Packets are processed by the Netfilter firewall if the firewall rules cannot fully be offloaded into NetFPGA because of limited memory or logic resource. The network stack utilizes the existing Linux network kernel, so currently our prototyping implementation does not require modifying the native Netfilter framework in the network kernel. Furthermore, next section, "multi-level traffic classification technique" will introduce work partition between software-based and NetFPGA-based firewalls in detail.

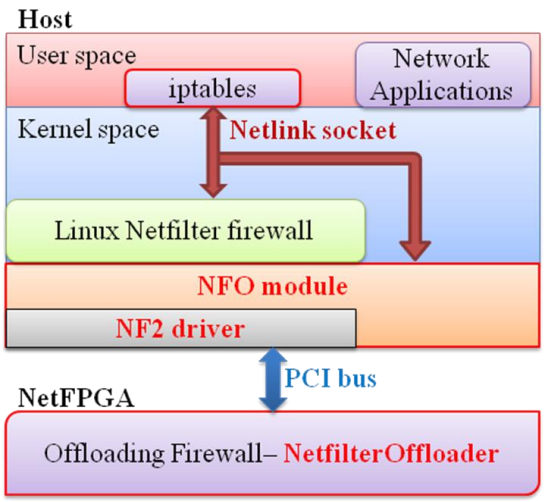


Figure 4: Software Architecture

4.2 Multi-level traffic classification technique

NetFPGA was optimized to be a low-cost teaching and research prototyping platform, and NetFPGA did not have enough FPGA resource and memory for deep content-level processing, according to our study. However, most malicious packets cannot be classified by only using the well-known ports; the Deep Packet Inspection (DPI) technique is required to identify malicious packets. As a consequence, we propose multi-level traffic classification architecture supporting both header-level and content-level classification techniques. NetfilterOffloader firewall in NetFPGA performs header-level packet classification and filtering, and Netfilter framework can be ported content-level classification functions, such as L7-filter [9] or DPI system [10].

5. PERFORMANCE EVALUATION

We designed experiments that compared both NetFPGA-based and software-based firewalls, and observed the impacts of web server for both types of firewall under high traffic environment. This section contains two parts: First, the experimental setup describes the experimental environment. Second, experimental results show profiling data and performance analysis.

5.1 Experimental Setup

The experimental environment is built as shown in Figure 5. The experiments employ the Apache web server [11] as network application and the *httperf* [12] as the web clients for generating http trace. Client machine also generates the ping flood traffic using NetFPGA packet generator [13] to disturb the normal traffic, and the ping flood generator does not occupy the CPU resource in the client machine. The software and hardware environments of server and client are listed in detail in Table 1 and Table 2.

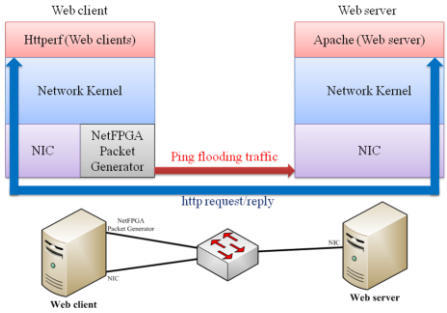


Figure 5: Experimental environment

Table 1: Hardware/Software environment of web server

Hardware	
CPU	Intel Quad Core
RAM	2 Gbytes
NIC	NetFPGA reference NICs vs. NetfilterOffloader firewall
Software	
Distribution	CentOS 5.4
Kernel	Linux kernel 2.6.18
Application	Apache 2.2

Table 2: Hardware/Software environment of web client

Hardware	
CPU	Intel Quad Core
RAM	2 Gbytes
NIC	Intel Corporation 82567LM-3 Gigabit Ethernet Controller
Packet generator	NetFPGA packet generator
Software	
Distribution	CentOS 5.4
Kernel	Linux kernel 2.6.18
Application	8 httperf clients

5.2 Experimental Results

This section presents two experiments for profiling the performance of web server equipping different types of firewalls. Besides, both Netfilter and NetfilterOffloader firewalls were inserted the rule that drops the ICMP echo request packets from specific source IP. The first experiment observes the performance of the both firewalls with growth of connection rate and fixed ping flood rate. Besides, we also measured the performance of non-flood attack situation for the Netfilter firewall. The second experiment describes experimental results with increasing ping attack packets and fixed connection rate.

5.2.1 Increasing the http connection rate

This experiment observes the throughput of the web server with the increasing connection rate and fixed ping flood rate at 50 Mbit/s. The profiling time of each connection rate is fixed at 60 s. In addition, we set the http client timeout within 1 s to avoid exhausting client resources. If the clients

cannot receive http reply within client timeout, those cases would belong to the client-timeout error.

The web server which is built with Netfilter firewall only has a reply rate below 1500 replies/s while the http connection rates increases above 3000 conns/s as shown in Figure 6. However, NetfilterOffloader firewall can still hold the reply rate of around 2500 replies/s, as shown in Figure 6. NetfilterOffloader curve performs little better than non-flood curve since it seems that non-flood situation still needs to compare every packet with firewall rule in Netfilter framework. However, NetfilterOffloader firewall has already offloaded the rule in NetFPGA so host kernel does not require classifying each packet. Additionally, the NetfilterOffloader firewall can reduce client-timeout errors about 1000 per second compared to the Netfilter firewall in high traffic rate, as shown in Figure 7. Those experimental results indicate that the NetfilterOffloader firewall can effectively block the ping flood attack as the non-flood situation in high traffic rate, so the host can reserve more CPU time for handling client requests rather than processing the attack traffic.

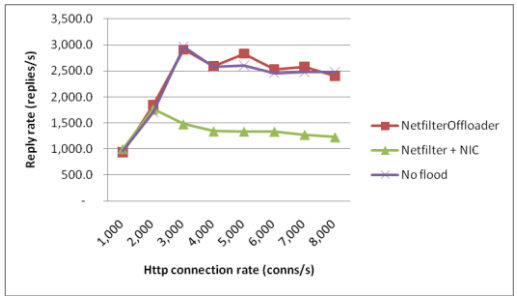


Figure 6: Http reply rate with the growth of the http connection rate

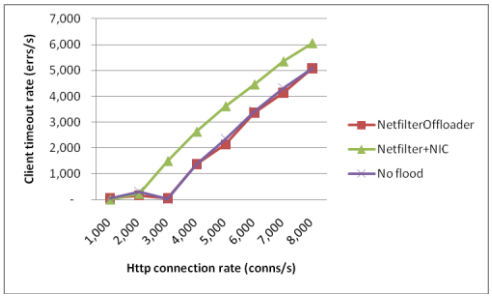


Figure 7: Client timeout error with the growth of the http connection rate

In addition, we profiled the http reply time between sending the http request and receiving the http reply in the low-traffic case. We extended the client-timeout to 10 s for reducing client time errors. From the measuring result for http reply time, NetfilterOffloader firewall can guarantee the http reply time less than 1 ms as the non-flood situation, as illustrated in Figure 8. As a consequence, packet filtering function offloaded into NetFPGA does not cause obvious extra latency and is capable of achieving the purpose of application isolation. Nevertheless, if a tremendous amount of attack traffic enters the host for the web server equipping Netfilter firewall, it brings extra interrupt handling and packet classification overhead to increase the responding time of network application. Furthermore, the reply time for Netfilter firewall increases up to 70 ms at 1100 conns/s as shown in Figure 8 because packet drop occurs and network kernel starts the TCP retransmission. The client-timeout errors become severe when the connection rate increases to more than 1400 conns/s, and the http reply time will become worse and unreasonable.

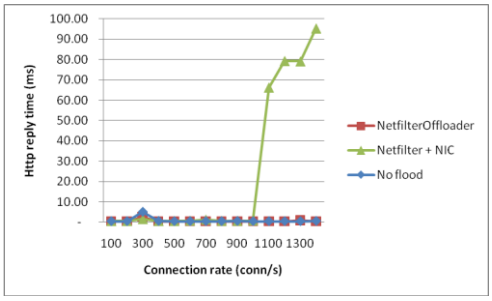


Figure 8: Http reply time with the growing connection rate in low connection rate

5.2.2 Increasing the ping flood rate

This experiment increases the flood rate for observing the performance of the web server on both NetfilterOffloader and Netfilter firewalls, and chooses the connection rate at 3000 conns/s and the client timeout at 1 s. With the rising of the ping flood rate, Netfilter firewall spends growing amount of time on filtering the ICMP echo request packets, and the performance of the web server declines, as indicated in Figure 9. Nevertheless, the NetfilterOffloader firewall can effectively protect the web server from ping flood attack, keeping the client timeout error rate below 500 packets/s, as illustrated in Figure 10.

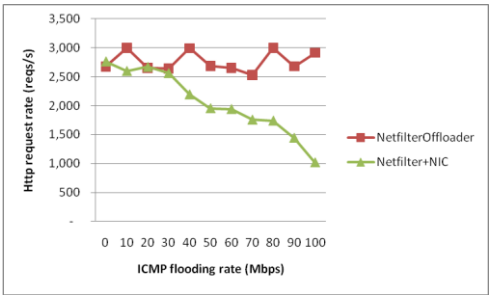


Figure 9: Http reply rate with the growth of the ICMP flood rate

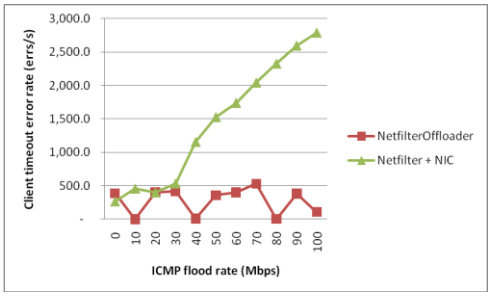


Figure 10: Client timeout error with the growth of the ICMP flood rate

We decreased the connection rate at 1000 conns/s with different flood rates, and increasing the client timeout to 10 s for profiling the http reply time and alleviating client-timeout errors. According to measuring result of http reply time, NetfilterOffloader firewall still holds a lower responding time than Netfilter firewall as shown in Figure 11, even in the low connections; the http reply time increases significantly above 80 Mbit/s flood rate also because of packet loss and TCP retransmission.

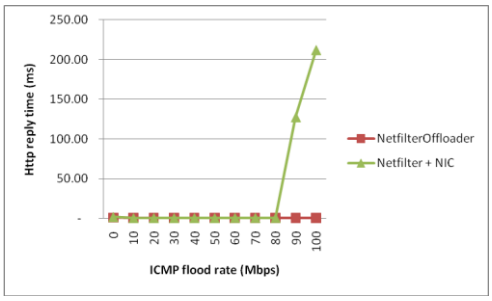


Figure 11: Http reply time with the growth of the flood rate in low connection rate

6. RELATED WORK

Many FPGA-based hardware acceleration systems move the host-end workload to the FPGA-end, to elevate the overall performance. In other words, the jobs are partitioned and distributed between hardware and software. For examples, Snort offloader adds pre-filter functions into FPGA to reduce the network-intrusion detection system (NIDS) - Snort's loading [14], and Shunt sheds loading of network-intrusion prevention system (NIPS) into NetFPGA [15].

Some researches utilized the network processors to accelerate protocol processing in the host. For instances, LRP implemented the early de-multiplexing on network processor [16]. Intel also developed the network processor to accelerate the Linux Netfilter firewall [17].

Our work focuses on giant server systems, e.g., web servers, real-time streaming servers, and deep packet analysis systems, etc, and offloading the Netfilter framework's partial functions into NetFPGA. The NetfilterOffloader firewall supports the early filtering and achieves the goal of application isolation, so as to improve the performance of the network-intensive application system.

7. CONCLUSIONS

In this paper, we designed and implemented a high-speed firewall: NetfilterOffloader firewalls on NetFPGA. Besides, the NetfilterOffloader firewall is integrated with the native Linux Netfilter firewall, and both types of firewalls are configured through means of *iptables* utilities. In addition, the NetfilterOffloader firewall can efficiently reduce the packet-filter burden in the host, early filter a tremendous amount of attack traffic and achieve the purpose of application isolation. According to the experimental results, we chose the web server as the example of the network-intensive application system. The web server equipping the NetfilterOffloader firewall can effectively prevent the attack traffic from affecting the web service, and has better throughput and response time.

8. FUTURE WORK

We will offload the connection tracking modules in the Netfilter framework into the NetfilterOffloader. A connection tracking module would collect connection information to support behavior-based classification. In addition, we will port the DPI system [10] into the Netfilter framework to support content-based classification. Besides, using the NetFPGA platform accelerates the DPI system.

9. ACKNOWLEDGMENTS

This research was financially supported by the National Science Council, Taiwan, Republic of China, under grants No. 98A063, NSC98-2219-E-006-002 and NSC98-2219-E-006-003, for which we are grateful.

10. REFERENCES

[1] H. Welte. "What is Netfilter/IPTables?" http://www.netfilter.org

[2] J. W. Lockwood, N. McKeown , G. Watson, G. Gibb , P. Hartke , J. Naous , R. Raghuraman , J. Luo, "NetFPGA - An Open Platform for Gigabit-Rate Network Switching and Routing," Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education, p.160-161, June 03-04, 2007

[3] J. Salim, H. Khosravi, A. Kleen, A. Kuznetsov, "Linux Netlink as an IP Services Protocol," RFC 3549, IETF, July 2003.

[4] J. Naous, G. Gibb, S. Bolouki, N. McKeown, "NetFPGA: reusable router architecture for experimental research," In PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow, pages 1–7, New York, NY, USA, 2008. ACM

[5] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," In Symposium On Architecture for Networking and Communications Systems, 2008 (ANCS '08).

[6] Xilinx Core Generator System - http://www.xilinx.com/tools/coregen.htm

[7] Xilinx. Xilinx Content-Addressable Memory v5.1 Product Specification, V 2.1, Nov. 11, 2004

[8] G. A. Covington, G. Gibb, J. Naous, J. W. Lockwood, and N. McKeown, "Encouraging Reusable Network

Hardware Design." International Conference on Microelectronic Systems Education, 25-27 July 2009

[9] L. Gheorghe, "Designing and Implementing Linux Firewalls with QoS using Netfilter, iproute2, NAT and l7-filter," PACKT Publishing , Oct. 2006

[10] C. S. Yang, M. Y. Liao, M. Y. Luo, S. M. Wang, "A Network Management System Based on DPI." In the Proceeding of the fourth international workshop on Advanced Distributed and Parallel Network Applications (ADPNA-2010). Takayama, Gifu, Japan, Sept. 14-16, 2010.

[11] Apache Team. "Apache HTTP server project." http://www.apache.org/

[12] D. Mosberge , T. Jin, "Httperf: A Tool for Measuring Web Server Performance," ACM, Workshop Internet Server Performance, pp. 59-67, June 1998.

[13] G. A. Covington, G. Gibb, J. W. Lockwood, N. Mckeown, "A Packet Generator on the NetFPGA Platform, " IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), April 2009.

[14] H. Song, T. Sproull, M. Attig, J. Lockwood, "Snort offloader: A reconfigurable hardware NIDS filter." In Proceedings of 15th International Conference on Field Programmable Logic and Applications (FPL), Tampere, Finland, Aug. 2005.

[15] N. Weaver, V. Paxson, J. M. Gonzalez, "The shunt: an FPGA-based accelerator for network intrusion prevention," Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays, February 18-20, 2007, Monterey, California, USA

[16] P. Druschel, B. Gaurav, "Lazy Receiver Processing (LRP): "A Network Subsystem Architecture for Server Systems." In Proceeding 2nd Symposium on Operating Systems Design and Implementation, Oct. 1996.

[17] K. Accardi, T. Bock, F. Hady, J. Krueger, "Network processor acceleration for a Linux Netfilter firewall," In Symposium on Architecture for Networking and Communications Systems, 2005.