I. J. Computer Network and Information Security, **2013, 10, 30-36**
Published Online August 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.10.05

Increasing the Efficiency of IDS Systems by Hardware Implementation of Packet Capturing

Zeinab Latifi, Kamal Jamshidi, Ali Bohlooli
Department of Computer Engineering, University of Isfahan, Isfahan, Iran
zeinab.latifi@yahoo.com, jamshidi@eng.ui.ac.ir, bohlooli@eng.ui.ac.ir

Abstract — Capturing is the first step in intrusion detection system (IDS). Having wire speed, omitting the OS from capturing process and no need for making a copy of packets from the system's environment to the user's environment are some of the system characteristics. If these requirements are not met, packet capture system is considered as the main bottleneck of IDS and the overall efficiency of this system will be influenced. Presence of all these three characteristics calls for utilization of hardware methods. In this paper, by using of FPGA, a line sniffing and load balancing system are designed in order to be applied in IDS systems. The main contribution of our work is the feasibility of attaching labels to the beginning part of each packet, aiming at quick easy access of other IDS modules to information of each packet and also reducing workload of these modules. Packet classification in the proposed system can be configured to 2, 3, and 5 tuple, which can also be applied in IDS detection module in addition to load balancing part of this system. Load balancing module uses Hash table and its Hash function has the least flows collisions. This system is implemented on a set of virtex 6 and 7 families and is able to capture packets 100% and perform the above mentioned processes by speed of 12 Gbit/s.

Index Terms — Packet capture, load balancing, hashing, IDS, FPGA

I. Introduction

Intrusion detection systems (IDS) by following user's activities would warn network administrators about attacks, security threats and any unauthorized access to perform appropriate security action [1].These systems have the following four major parts:

1. Packet Capturing: sniffing the line and capturing network packets and delivering them to higher layers
2. Pre-processing: Stateful inspection of packets along with assessing each packets for checking the handshakes in a communication session by using the rules of the communication protocols and also creating state machine in order to identification of current event and prediction of next event [2]
3. Detection: detection of attacks by using Signature-Based, Anomaly-Based and Stateful Protocol Analysis methods [1].
4. Warning: warning and announcing the traffic characteristics of the attacker

In this paper we try to design and implement a developed packet capture system for enhancing the efficiency of IDS. In the following, the general points about this packet capture system will be explained.

The packet capture system first receives raw traffic of the network from a physical Ethernet interface and then performs the first needed processing on it. Captured packets must be copied in hardware memory in order to be transferred to the memory of operating system by using an interface. Afterward, it is the operating system's job to deliver the packets to the applications [3].

The next part of IDS is pre-processing of the packets; thus, in order to decreasing operating system's tasks in this module and increasing IDS efficiency, it is better for a line sniffing system performs some initial packet processing than its normal jobs. Decoding, classification and load balancing of packets between processor cores or available processors are parts of processing which this system does besides packet capturing. Hardware implementation of this processing will prevent from wasting time in the operating system and affects IDS speed greatly.

Because the network based IDS collects raw packets from the line, it must repeat all network stack operations that were performed in the OS level in order to transfer data to the application. Among these operations, decoding will be performed in the capturing module. Decoding means acquiring different fields of packets. This operation will be discussed in section 3 in depth.

Classification of packets in capturing module is performed with the goal of separating packets based on their flows. Flow is a collection of packets which has some similar fields in their headers. In fact, assessing flows will deliver more information to the IDS than assessing each packet separately. Also for detection of most anomalies it is needed to asses a collection of packets [2, 4]. In this phase, for each flow an entry is allocated in the lookup table and also a key is consider for that. This key is obtained by some fields of packet header. When a new packet enters, available entries in the memory are searched and if a similar flow to the packet flow is found, that entry will be updated. If no, a new entry must be created in the table.

Another duty of packet capture system is load balancing among different applications and actually different processor cores or available processors. Because the speed of today networks links is increased and efficiency of processors does not satisfy this speed, capturing traffic of the network must be divided into processable divisions in order to reduce workload of each processing engine and increase processing speed. This aim could be accomplished by creating separate queues for packets with regard to their flows and allocating each queue to one processor core [5].

The aim of this article is to design a packet capture and load balancing system, which in addition to having ability to sniffing line by 12 Gbit/s; it can reduce workload of other sections of IDS and accelerate their speed by its added capabilities. The first capability of the proposed systems is label attachment to the beginning part of each packet. These labels are in fact, required data for the pre-processing module that are acquired by re-decoding packets in this module before by software methods. Adapting this capability will result in prevention of doing repeated actions in this module, reduction of workload and also an increase in speed. Feasibility of triple configuration of load balancing module is formulating another capability, which in the case of appropriate configuration; this capability will provide opportunity to omit packet classification process from IDS detection module and consequently will accelerate its operation speed.

The remainder of this paper is organized as follows. In Section 2 a review is conducted on the previous works focused on different packet capturing methods in software and hardware. Furthermore, packet classification and load balancing methods are mentioned as well. Section 3, sniffing system is discussed in detail and a proposed method for its improvement is explained. Implementation of system on FPGA and its results is available in Section 4 and finally Section 5 is the conclusion of the article which also provides suggestions for future researches.

II. Related Works

The speed of different network connections and initial processing conducted in the packet capture module; result in variation in packet capture techniques. These methods are divided into hardware and software groups. In the following some of these techniques are discussed.

In [3] a software method is provided .In this method, for accelerating packet capture, a type of special socket is introduced that is based on a circular buffer where incoming packets are copied and by doing so, it could reduce CPU time to capture and transfer packets.

In [6] for speeding up the capturing process, DNA drives are used. These drivers by using processor unit of network cards and also DMA rings would accelerate capturing and transferring packets to the operating system. These drivers exist in 1 gbit/s cards base on Intel (el000e family) and also 10 gbit/s cards (82598/99).

In [7, 8] a method named TNAPI is presented which uses I/O acceleration technique and PF_RING. This method, other than to packet capturing, performs load balancing between some applications and now, is one of the best software methods used.

In [9] nCAP is introduced that it could capture packets in wire speed by upgrading hardware driver and using a kernel module. Although this method claims the direct transfer of packets to the application layer, but it seems that this method is not followed by designers for permanent use in Linux cores.

Hardware methods of packet capture adapt with lower layer of network stack and also copy packets directly from network card to the user environment, without interference of operating system. So, usage of these methods speeds up the capturing process tremendously. For this purpose, in [10] DAG hardware is introduced. Ability to capture packets in speed equal to 10Gb/s, load balancing, packet filtering, traffic replication and attaching time stamps are some of DAG's characteristics.

In [11, 12] another hardware, CAMBO, is delivered. Combo cards include basic and interface cards which perform network data processing like packet capturing in 10 Gbit/s by transferring complex algorithms on FPGA. Of course, this card has got higher speeds by increasing number of its ports.

Another hardware method is NetFPGA which was referred in [13, 14]. This method is a flexible and simple tool for pattern matching. Also a part of its designing is dedicated to capturing and classifying of packets by wire speed, additionally filtering of packets is done by software methods too. This method uses virtex-II Pro 50 FPGA.

Due to the diversity of packet classification and load balancing methods, the next section will review the available methods. Generally, with respect to implementation framework, packet classification will be categorized into two methods: algorithmic and architectural. In algorithmic methods, it is tried to make packet fields processing faster by using available search algorithms. But these methods have no high efficiency. As a result, architectural methods have solved this problem by using high processing speed in hardware [15].Ternary content-addressable memory (TCAM) is one of the famous hardware methods which is used for parallel high speed search. But low efficiency, high cost, high power consuming and limitation in scalability for long keys [16] are shortcomings of this method.

In [15] a comprehensive survey is performed on packet classification algorithms. Exhaustive search, decision tree, field decomposition and tuple space are among these algorithms. But due to use of prefix matching of addresses in these algorithms and lack of access to address prefixes in packet capture level, these methods cannot be used. On the other hand, in most articles which needed fast exact match, Hash based methods are used. Compare to available algorithms, this method has less complexity and more efficiency. But the main problem of this method is collision probability during inserting elements. Different methods are proposed to face this problem in [17-19].

III. The Proposed Developed Packet Capture System

Block diagram of the proposed packet capture system is shown in Fig. 1. Input of this system is the output traffic of a 10 Gb Ethernet interface and its output are packets which are in 8 separate memories related to each processor. Different parts of this system are illustrated as following.

A. Ethernet Module

The first step of packet capture system is Ethernet module. Frame decoding, separating their body from header and delivering frame body to the next module are the main duties of this module. By decoding each frame, some information such as source and destination MAC addresses and frame type will be identified. This information is conducted to some parts of IDS which needs them. Another duty of this module is to set flags which are added to each packet in order to identify frames with incorrect CRC.

B. IP Header Extractor Module

Because most attacks are on IP packets, proposed packet capture system only focus on IP packets. IP Header Extractor Module by decoding packets obtains source and destination IP addresses, datagram length, protocol type and offset of divided packets. Some of this information is delivered to next IDS modules as output to avoid repeated decoding. Then IP packets bodies with respect to the load balancing output are transferred into the memory of one of the processors.

C. Load Balancing Module

Classifying packets based on their flows is the first stage of load balancing module. Separating flows is done based on matching of some of the packet header fields. In DAG card which was explained in Section 2, only source and destination address IP fields are used. For load balancing in packet capture stage these two fields are sufficient. But other modules of IDS like signature-based detection module needs packet classification based on more fields [20]. So, in the proposed method for decreasing workload of these modules, configuration capability is added to the load balancing module. In fact, this system can be configured in 3 different types that include of 2, 3 and 5 tuple. In 2 tuple classification, matching of source and destination IP fields are considered, while in 3 tuple, protocol field is added to previous fields. In 5 tuple classification, destination and source port fields are added as well. This quality results in more flexibility of proposed packet capture system.

Considering comparisons in this paper, hash based algorithms are used for classifying packets. In hash based algorithm of proposed system, first a lookup table is created for flows so that each of its entry is related to a flow. Then hash of considered fields of packets is calculated and the packet is allocated to an entry with hash value index. Output of each entry is the queue number that each packet must be placed in. Since some of hashing collision reduction methods are not synthesizable and they reduce the speed of system [18], a hash function is used in the proposed system that can minimize the number of collisions. The remaining collisions can be ignored. Suggested function is error detection CRC function.

D. Write Controller Module

This module takes output of load balancing module which is the queue number of each packet and prepares data which must be written in that queue. As previously mentioned, one of the capabilities added to the proposed system is attaching of labels to the beginning of each packet. These labels are extracted from each packet by Ethernet and IP header extractor modules and are delivered to this part as input. Ethernet frame type, IP addresses, important offsets in packet, flags for warning about CRC error and checksum are among such labels. This capability provides the opportunity for other IDS parts like pre-processing to access this information without any extra processing and also omits some processes like decoding which was carried out by software in some IDS like Snort [21].

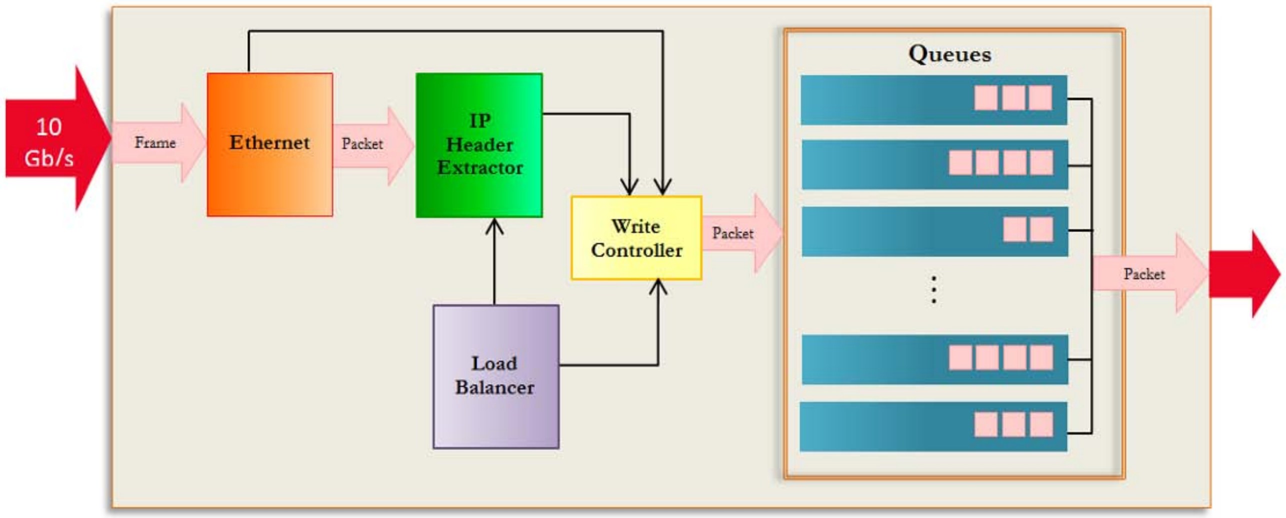


Figure 1: Block diagram of system

E. Queues

In this proposed packet capture system, 8 separate FIFO queues are considered for 8 processor cores or different processors. The number of queues in this system can be increased easily and needs no extra processing. Load balancing module by connection with this part will be informed about the size of queues. After that, this module will choose a queue with least occupied volume during selecting queue for initial packets of each flow. Next packets of that flow will be located in the same queue.

IV. IMPLEMENTATION AND RESULTS

The proposed method for packet capture has been written in VHDL language and Active HDL 8.2 has been used for simulation. Accuracy of this system has been measured by real traffic of the network in simulation level.

Fig. 2 shows the simulation result of Ethernet module. For more clarity, we have shown results for one packet. Ethernet module separates frames payload from their header and delivers them to the next module. The value of Header fields such as frame type and also frame validity are determined by this module. As seen from Fig. 2, start and end of each frame are other outputs of this module.

Simulation results of IP Header Extractor module is shown in Fig. 3. When a new packet arrives, this module starts it's decoding. As seen from Fig. 3 header length of current packet is 5 and its datagram has 40 bytes length. Fragment flags, IP addresses and protocol type are other header fields that this module has extracted from this packet. By finishing the header data, IP Header Extractor has collected the entire packet payload parts in one output signal named *packet* and has sent it to the next module. Also, Complete arrive time of packet has informed by this module too.

Fig. 4 shows Load Balancing module simulation results. As mentioned before, this module determines the appropriate queue for each packet according to the configuration type. In hardware description step of the proposed system, for more simplicity Write Controller module is considered as a part of Load Balancing. So, attaching of labels to the beginning of each packet is other duty of Load Balancing module. Therefore, needed information of labels is given to this part in addition to classification information.

Fig. 5 shows lookup table of Load Balancing module. As seen hash value of current packet is 88H. So, the number of appropriate queue that is 8 has written in this entry of lookup table. The memory of queues is shown in Fig. 6. For showing the accuracy of result we use only one packet for simulation.

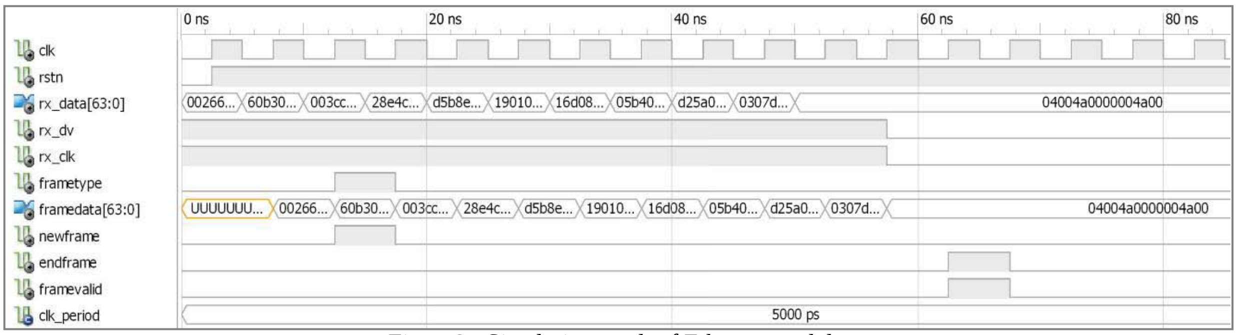


Figure 2: Simulation result of Ethernet module

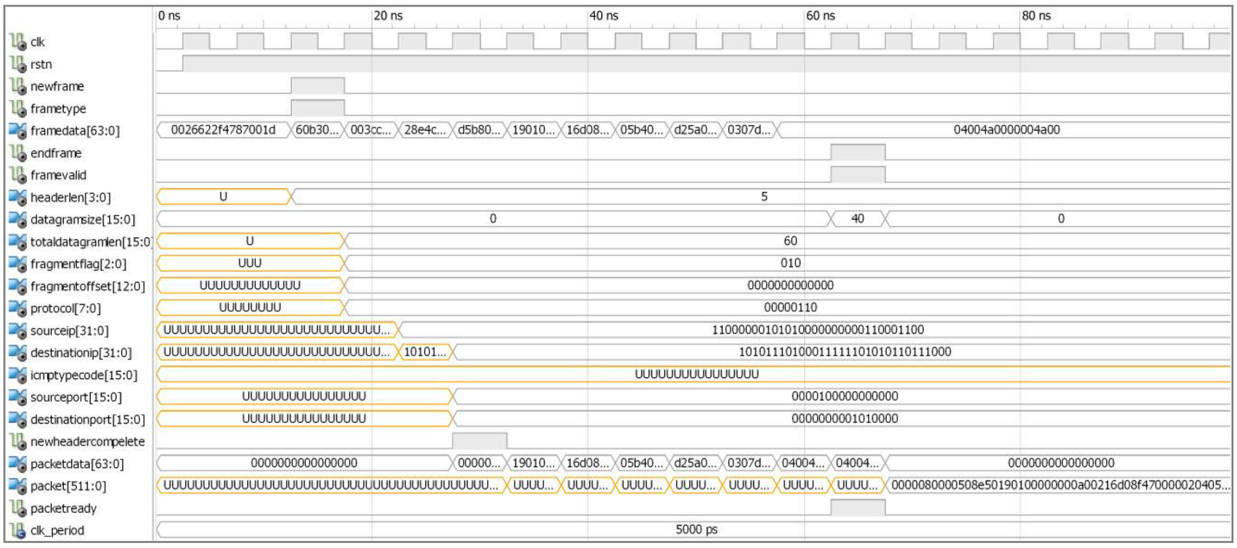
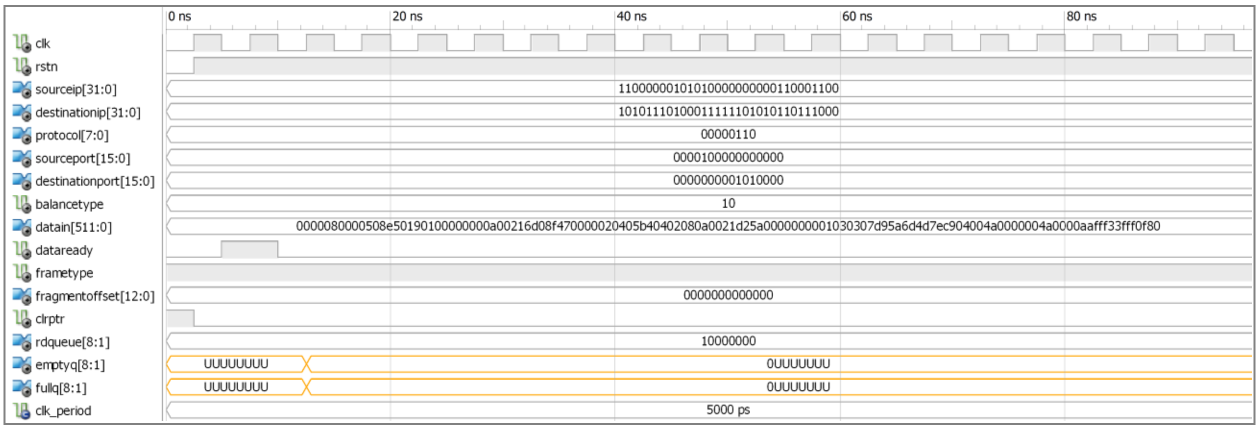
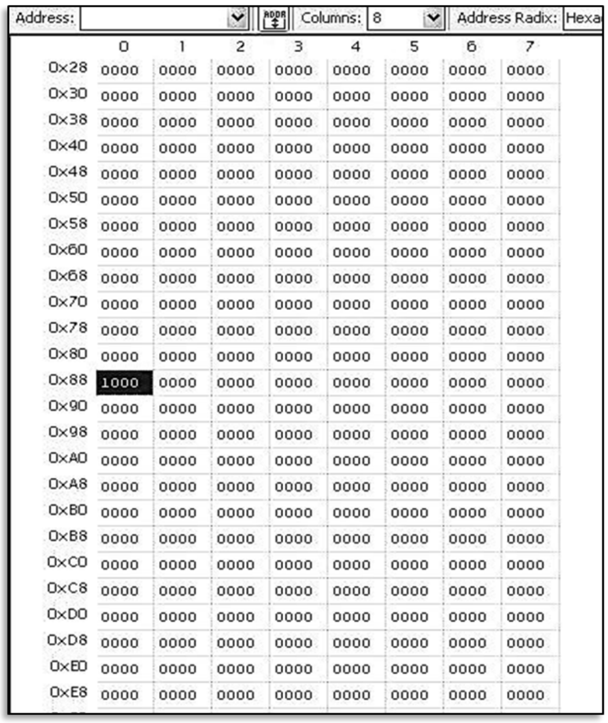


Figure 3: Simulation result of IP header extractor module



```
                      0 ns          20 ns          40 ns          60 ns          80 ns
clk
rstn
sourceip[31:0]        11000000101010000000000110001100
destinationip[31:0]   10101110100011111101010110111000
protocol[7:0]         00000110
sourceport[15:0]      0000100000000000
destinationport[15:0] 0000000001010000
balancetype           10
datain[511:0]         0000080000508e50190100000000a00216d08f470000020405b40402080a0021d25a0000000001030307d95a6d4d7ec904004a0000004a0000aafff33fff0f80
dataready
frametype
fragmentoffset[12:0]  0000000000000
clrptr
rdqueue[8:1]          10000000
emptyq[8:1]           UUUUUUUU      0UUUUUUU
fullq[8:1]            UUUUUUUU      0UUUUUUU
clk_period            5000 ps
```

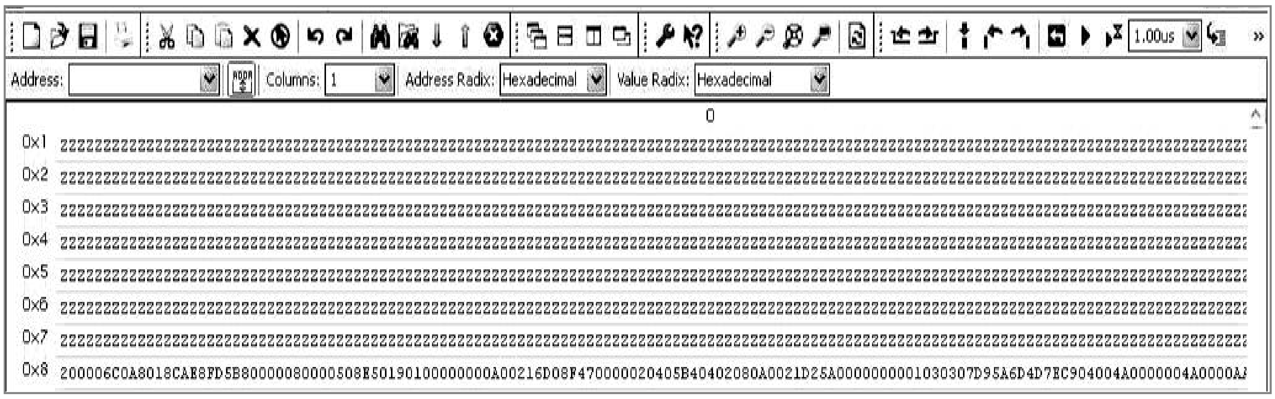
Figure 4: Simulation result of load balancing module



Address: Columns: 8 Address Radix: Hexa

	0	1	2	3	4	5	6	7
0x28	0000	0000	0000	0000	0000	0000	0000	0000
0x30	0000	0000	0000	0000	0000	0000	0000	0000
0x38	0000	0000	0000	0000	0000	0000	0000	0000
0x40	0000	0000	0000	0000	0000	0000	0000	0000
0x48	0000	0000	0000	0000	0000	0000	0000	0000
0x50	0000	0000	0000	0000	0000	0000	0000	0000
0x58	0000	0000	0000	0000	0000	0000	0000	0000
0x60	0000	0000	0000	0000	0000	0000	0000	0000
0x68	0000	0000	0000	0000	0000	0000	0000	0000
0x70	0000	0000	0000	0000	0000	0000	0000	0000
0x78	0000	0000	0000	0000	0000	0000	0000	0000
0x80	0000	0000	0000	0000	0000	0000	0000	0000
0x88	1000	0000	0000	0000	0000	0000	0000	0000
0x90	0000	0000	0000	0000	0000	0000	0000	0000
0x98	0000	0000	0000	0000	0000	0000	0000	0000
0xA0	0000	0000	0000	0000	0000	0000	0000	0000
0xA8	0000	0000	0000	0000	0000	0000	0000	0000
0xB0	0000	0000	0000	0000	0000	0000	0000	0000
0xB8	0000	0000	0000	0000	0000	0000	0000	0000
0xC0	0000	0000	0000	0000	0000	0000	0000	0000
0xC8	0000	0000	0000	0000	0000	0000	0000	0000
0xD0	0000	0000	0000	0000	0000	0000	0000	0000
0xD8	0000	0000	0000	0000	0000	0000	0000	0000
0xE0	0000	0000	0000	0000	0000	0000	0000	0000
0xE8	0000	0000	0000	0000	0000	0000	0000	0000

Figure 5: Lookup table



Address: Columns: 1 Address Radix: Hexadecimal Value Radix: Hexadecimal

```
                                   0
0x1 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x2 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x3 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x4 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x5 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x6 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x7 zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
0x8 200006C0A8018CAE8FD5B80000080000508E50190100000000A00216D08F470000020405B40402080A0021D25A0000000001030307D95A6D4D7EC904004A0000004A0000A
```

Figure 6: Queues

TABLE I. IMPLEMENTATION RESULTS ON SOME DIFERENT FPGAS

	Xc6vhx565t	Xc6vhx255t	Xc6vhx380t	Xc7v485t	Xc7v585t	Used
Slice Registers	6%	15%	10%	8%	6%	49079
Slice LUTs	4%	10%	7%	5%	4%	17093
Fully Used LUT-FF Pairs	0%	0%	0%	0%	0%	297
Bonded IOBs	27%	36%	27%	29%	20%	175
Block RAM/FIFO	0%	1%	1%	0%	1%	8
BUFG/BUFGCTRLs	21%	21%	21%	21%	21%	7
Maximum Freq. (MHz)	157.594	173.295	173.295	192.879	192.879	
Speed(Gb/s)	10.086	11.090	11.090	12.344	12.344	

For synthesis of written codes, XST synthesis tool has been applied and for implementation VHDL cods on FPGA, ISE 13.1 software has been used. This system has been implemented and tested on a FPGA which has10 Gb/s Ethernet port. Comparison of important parameters of implementation on different chips, such as maximum frequency, final speed, utilization of slice Registers, slice LUTs and etc. is shown in table 1. The results show Xc7v585t has best speed and consumed fewer resources.

VI. CONCLUSIONS

In this article a line sniffing system and load balancing for being used in IDS was designed and implemented. The aim of this system is 100% capturing of packets and accelerating function of next parts of IDS. For doing so, labels are attached to the beginning of the packets in order to prepare needed data for the other parts of IDS, e.g. Pre-processing. Also by suitable configuration of load balancing module, this system can classify packets in a way that signature-based detection module do not need to repeat it. Adding these characteristics to the system has no negative effect on the main role of the system. The proposed method has been implemented on Virtex6 and 7 families and it is able to capture packets in 12 Gbit/s and perform the mentioned processing.

REFERENCES

[1] K. Scarlone and P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS).U.S: NIST, 2007.

[2] S. Li, J. Torresen and O. Serrisen, "Improving a Network Security System by Reconfigurable Hardware," in Proc. 2004 Norchip Conf. ,Oslo, Norway, pp. 135-138.

[3] L. Deri. "Improving Passive Packet Capture Beyond Device Polling," in 4th Int. Conf. System Administration and Network Engineering (SANE), Amsterdam, The Netherlands, 2004.

[4] W. Jiang and V. Prasanna, "Scalable Packet Classification on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. ,vol. PP, pp. 1 – 13, June 2011.

[5] J. Wang, Y. Xie, Ch. Zhu, Z. Zhao and Ch. Han, "An Embedded Load Balancing System for High Speed OC192 Networks," in 2009 Int. Conf. Embedded Software and Syst. , Zhejiang, pp. 587-592.

[6] "Introducing PF_RING DNA (Direct NIC Access)," Feb. 21, 2010. [Online]. Available:http://www.ntop.org/pf_ring/introducing-pf_ring-dna-direct-nic-access.html.[Accessed: 10 November 2011].

[7] L. Deri and F. Fusco. "Exploiting Commodity Multi core Systems for Network Traffic Analysis," July 2009.[Online]. Available: http://luca.ntop.org/MulticorePacketCapture.pdf [Accessed: 10 November 2011].

[8] "Intel ®I/O Acceleration Technology," Aug. 29, 2011. [online]. Available: http://www.intel.com/content/www/us/en/wireless-network/accel-technology.html. [Accessed: 10 November 2011].

[9] L. Deri, "nCAP: wire-speed packet capture and transmission," in 2005 Workshop on End-to-End Monitoring Techniques and Services, pp. 47-55.

[10] "Enterprise Network Monitoring Tools - Network Security System - Application Performance Monitoring," 2005 [online]. Available: http://www.endace.com/dag-high-speedpacket-capture-functions-and-features.html. [Accessed: 15 October 2011].

[11] J. Novotný, M. Žádník, "COMBOv2 - Hardware Accelerators for High-Speed Networking," 2008. [online]. Available: http://www.cesnet.cz. [Accessed: 20 October 2011].

[12] "COMBO Cards - Development and Commercial FPGA boards," 2008. [online]. Available: http://www.invea-tech.com/products-and-services/combo-fpga-boards. [Accessed: 20 October 2011].

[13] M. Scott, "A Wire-speed Packet Classification and Capture Module for NetFPGA," in First European NetFPGA Developers, Cambridge, UK, 2010.

[14] J. Shafer, "NetFPGA Hardware Architecture," Des. 2008. [online]. Available:

http://comp519.cs.rice.edu. [Accessed: 20 October 2011].
[15] D. E. Taylor, "Survey & Taxonomy of Packet Classification Techniques," J. acm Computing Surveys(CSUR), vol. 37, pp. 238 – 275,Sep. 2005.
[16] Yu Fang, R.H. Katz and T.V. Lakshman, "Gigabit rate packet pattern-matching using TCAM," in Proc. 12th Int. Conf. IEEE Network Protocols (ICNP), 2004, pp. 174 - 183.
[17] M. Singh and D. Garg, "Choosing Best Hashing Strategies and Hash Functions," in 2009 IEEE Int. Conf. Advance Computing (IACC), pp. 50-55.
[18] K. Pati, H. Cheng, H.Sunt Kim and H. Ali Agha, "Data Structures and Algorithms-Topic 12: Hashing by Changing," May 4, 1999. [online]. Available: http://cgm.cs.mcgill.ca/~hagha/topic12/topic12.html. [Accessed: 10 November 2011].
[19] S. Kumar, J. Turner and P. Crowley, "Peacock Hashing: Deterministic and Updatable Hashing for High Performance Networking, " in Proc. 27th IEEE Conf. comput. Commun. , 2008, pp. 101-105.
[20] H. Song and J.W. Lockwood, "Efficient Packet Classification for Network Intrusion Detection using FPGA, " in Proc. 13th ACM/SIGDA Int. Symp. Field programmable gate arrays (FPGA),New York,2005, Pages 238-245.
[21] M. Roesch,"SNORT - lightweight intrusion detection for networks, "in 13th USENIX Conf. Systems Administration, Berkeley, 1999, pp. 229-238.

Zeinab Latifi received the BS degrees in Computer engineering from the University of Isfahan, Iran in 2010. She is now completing the MSC degree at the University of Isfahan, Iran. Her research interests include intrusion detection as well as FPGA based system design.

Kamal Jamshidi received the MS and PhD degrees in electrical engineering from Anna University of India in 1990 and I.I.T University of India in 2003, respectively. He currently is an assosiate professor in the Engineering Department of University of Isfahan. His research interests include wireless sensor network as well as fuzzy systems and microprocessor based systems.

Ali Bohlooli received the BS and MS degrees in Computer engineering (with honors) from the School of Electrical & Computer Engineering, Isfahan University of Technology, Iran in 2001 and 2003, respectively. He received the Ph.D. degree at the University of Isfahan, Iran in 2011. Now he is an assistant professor at department of computer engineering university of Isfahan, Iran. His research interests include wireless network and as well as hardware design.