Network Recorder and Player: FPGA-based Network Traffic Capture and Replay

Siyi Qiao, Chen Xu, Lei Xie, Ji Yang, Chengchen Hu, Xiaohong Guan, Jianhua Zou
Department of Computer Science and Technology School of electronic information and engineering
Xi'an Jiaotong University
Xi'an 710049, China
{qiaosiyi.900305, xuchen_224}@stu.xjtu.edu.cn, yangjifaust@gmail.com,huc@ieee.org,{xhguan, jhzou}@sei.xjtu.deu.cn

(Demonstration Paper)

***Abstract*— An appropriate tool to generate real network traffic plays an important role in testing network system. Traditionally, such a tool relies on software solutions that copies data back and forth between different part of memory to capture or replay network traffic. In this paper, we propose an FPGA-centric approach using parallel logic, which can ensure high accuracy of time and high throughput. We first design an FPGA add-on board dealing with the multifarious work like adding content or calculate statistical value. The system is implemented on an own designed off-the-shelf FPGA network add-on card to demonstrate the viability of our assumption. Experiments demonstrate reasonable performance improvement (higher throughput and replay time precision) when compared with software based solutions.**

Keywords—traffic capture, traffic replay, FPGA

I. Introduction

Internet traffic capture and replay is one of fundamental requirements in many internet communications. The system (e.g. Intrusion Detection System & Traffic Recorder) of internet security always needs the technology of backbone and ACESS traffic capture and replay. With rapid development of network technology and unprecedented increasing network requirement, researchers focus on the network security, the optimization of network performance and reasonable using of network resources [1]. The technology of Traffic-Replay can help to detect the compressive capacity of any internet system.

Paper [2] presented a network analytical method which needs a high-throughput and time-sensitive traffic capture and replay equipment. Commonly used network traffic tools mostly are Software capture tools (Sniffer, Tcpdump [3], Wireshark, etc.), Software replay tools (Tcpreplay) and many-core smart card. For capturing packet data, software tools need to copy packet data between memory spaces. They occupy lots of CPU Memory resource in the process of packet capturing. And software traffic replay tools are difficult to ensure packet throughput and time accuracy. The wireshark that runs on a general PC(i3,8G,1T) will drop 65% packets at the rate of 1Gbps. A many-core card just solves the low packet throughput shortcoming, while it cannot solve the problem of time accuracy dispersion. Moreover, all of the above tools are lack of scalability.

The research presented in this paper is supported in part by the NSFC (61272459, 61221063, U1301254), 863 High Tech Development Plan (2012AA011003, 2013AA013501) 111 International Collaboration Program, of China, Research Plan in Shaanxi Province of China (2013K06-38), and Program for New Century Excellent Talents in University(NCET-13-0450).

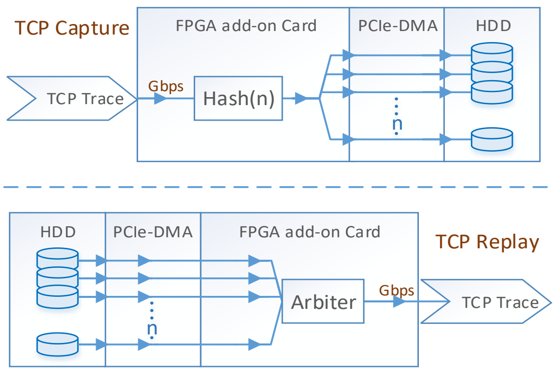

Fig. 1. Capture and Replay process schematic

Therefore, we present a method of packet capture and replay using FPGA parallel logic [4], which can ensure high accuracy of time and high throughput, just like the NetFPGA [5]. Using our own designed board which carries the network interface, we divide one high speed network traffic into many queues, at the same time adding user packet information in the packet header (Fig. 1). The add-on card send packet data to PC in DMA mode through the PCIe interface and then the data will be processed by software. In order to ensure the order of processing, we add a timestamp to each packet. When we try to replay the capture traffic, the system fetch packets from the four disks and then send them to the add-on board through PCIe. After comparing the timestamp, packets are sent out by the Ethernet port on the expansion card.

In the next section, we elaborate on the demonstration overview of the proposed system and then evaluate the performance in Section III. We make conclusions in Section IV.

II. DEMONSTRATION OF THE NETWORK TRAFFIC CAPTURE AND REPLAY SYSTEM

A. Architecture overview

In this paper, we proposed an improved network traffic capture and replay system based on the combination of hardware and software with high performance, such like efficiency, flexibility and scalability. In order to promote the data processing speed, we replace some traditional program software by an FPGA-centric network traffic process add-on board.

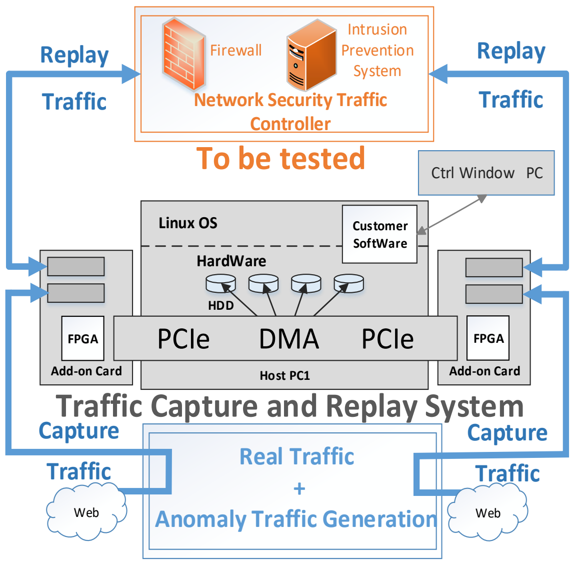

Fig. 2. application scenario experiment framework

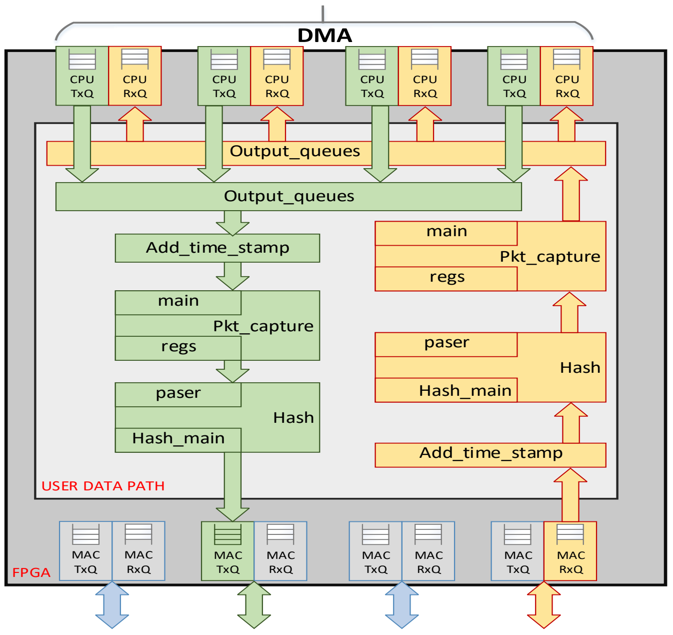

Fig. 3. application scenario experiment framework

As shown in Fig. 2, the part of the gray is the traffic capture and replay system in network test application scenario [1]. The logic includes network protocol conversion, user logic (UDP, User Data Path) and data conversion running on the add-on board. The core logic of capture and replay function is implemented inside the UDP module. The two pieces of add-on card controlled by the HOST will capture and replay the real traffic and anomaly traffic to the tested system. We adopted 4 data process queues for evaluation.

As shown in Fig. 3, in UDP module, the process shown in the right column that data up flow to the host is on behalf of capture process, while the process shown in the left that data down flow to the network interface is on behalf of replay. In traffic capture subsystem, packets can be captured from MAC (at a rate of Gbps), processed by hardware, hashed according to five-tuple, and then distributed to four queues, which ensures that all flow in the same session are assigned to one hard disk. In the meantime, the information of arrival time and length of packets is calculated in the hardware, transported to host in DMA mode and stored in disks along with packets, and then be processed by software. In traffic replay subsystem, flows that captured by capture subsystem will be load from four disks, outputted with the order of capture according to the timestamps, thus simulating real network traffic.

B. *The structure of traffic capture and replay card*

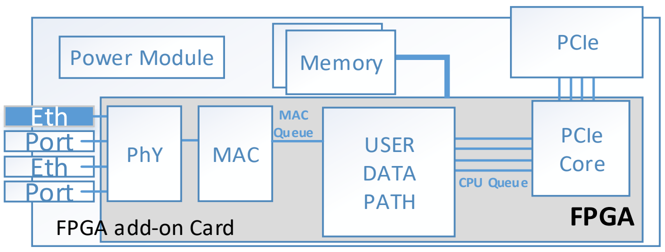

Fig. 4. the structure of traffic capture and replay card

Fig. 4 shows the structure of our own designed board. This PCIe expansion card consists of network interfaces, a FPGA chip and a PCIe interface, external memory and power modules. The following details are the performance of the specific device.

1) *Network Interface*
The expansion card has four SFP+ interface, supporting 1G/10G GBIC module.

2) *FPGA*
We choose XILINX XC7K325T-2FFG900 as major digital logic component, which has GTX ports with high performance, GTX is a class of IO interface owned by Xilinx FPGA, and they have the capability of high-speed serial transceiver.

3) *PCIe*
PCIe Bus is the main channel that links the FPGA and host, packets captured are transported by PCI-E bus.

4) *QDRII+SRAM*
The CY7C25632KV18 is Synchronous Pipelined SRAM, equipped with QDR II+ architecture.

C. *FPGA logic design*

1) *TRAFFIC CAPTURE (TC) ARCHITECHTURE*

Capture system processe packets received from one Ethernet interface and saved them to four queues through DMA. The procedure is divided into four stages, as shown in Fig. 3.

a) Add_timestamp

A time counter is implemented in this module to record packets arriving time and then attach every incoming packet with a timestamp header of 64 bits, further transmitted it to software as part of packets.

b) Hash

In this stage, TC parses each packet and extracts the five-tuple information, which is 104 bits combined by source IP, destination IP, source port, destination port and IP protocol type. TC will separate received traffic to four CPU queues based on five-tuple, Hash module balance traffic among CPU queues, and further guarantee that packets belongs to the same session are restored in the same queues.

c) Packet_capture

In addition to add timestamp header, software needs extra information of received packets, such as packets length. Therefore, in this module TC counts these info and encapsulate them into packet. It also calculates total captured packets numbers, length and rate, then records them in registers that can be read by software.

d) Output_queues

Since TC received traffic from one Ethernet interface and distributed them into four CPU queues, in this module packets are allocated to appropriate output queues according to output port calculated in Hash module.

2) *TRAFFIC REPLAY (TR) ARCHITECHTURE*

As shown in Fig. 3, TR received packets from four CPU queues which are stored in hard disks by TC steps and then aggregated them and send out from one Ethernet interface.

a) Add_ctrl_header

Since timestamp header is stored as parts of packet, TR need to identify this field from packet and in this module we add control flag to the timestamp header.

b) Arbiter

Packets send out from four CPU queues simultaneously are aggregated into one queue, TR needs to arbiter among the input queues according to timestamp header attached with each packet. It processes packets in the order of timestamp to make sure traffic replayed in the same sequence as packets are captured in TC.

c) Delay

Data processing rate is 8G in FPGA UDP pipeline, while sometimes real traffic rate is not as high as that. To replay traffic in a relative real situation, TR need to control time interval between packets based on the timestamp header. This module calculate every packet delay and only transmit them in specific time.

d) Packet_statistic

In this module, TR records the number, length and rate of packets sent from CPU queues in registers, which can be read by software. And since packets from CPU queues are attached with extra timestamp header that cannot be parsed by network card, we remove those headers from packets before output queue in Ethernet interface.

III. EVALUATION

A. *EXPERIMENT OF THROUGHPUT*

1) *Test environment*

PC1: The Add-on Card is installed on this host with Ubuntu 12.04, running TC/TR project and interact with host through PCIe interface.

Smartbits: It can send and receive packets from 4 ports (COP0~COP3) which are connected with 4 Ethernet interfaces (MAC0~MAC3) on The Add-on Card. Smartbits is controlled by another PC1 to set traffic rate. (Since 4 CPU queues are similar to 4 Ethernet queues, we use Ethernet interface to replace CPU queues and do loopback test using Smartbits.)

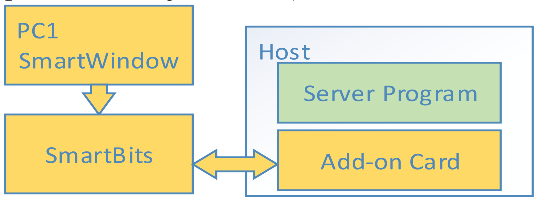

Fig. 5. System Test organization of the proposed framework

2) *TC/TR TEST*

a) Traffic Capture

We use Smartbits sending 127,503,023 packets of 64 bytes to Ethernet interface 0 on The Add-on Card. The result shows that these packets are distributed into four queues, respectively 31,875,756, 31,875,755, 31,875,756, 31,875,756 from queue 0 to 3, indicating the traffic balance among four queues. The statistic results recorded by hardware registers are consistent with Smartbits. Besides, we also verify that TC can allocate packets in the same session to one specific output queues.

b) Traffic Replay

We use the 4 ports of Smartbits to send packets to 4 Ethernet interfaces on The Add-on Card, which is equivalent to packets replaying from 4 CPU queues. We replay packets from 4 queues and aggregate them to one queue. Packets numbers from queue 0 to 3 are respectively 44,160,639, 44,160,639, 44,340,794, 44,316,028, and the total number of packets that replayed to network is 176,978,100. The results indicate that TR can aggregate traffic from 4 queues and replay packets from one Ethernet interface in line rate **without** packets loss.

3) SCALABILITY TEST

We make 3 kinds of inter logic about our project, and they owned 1, 2 and 4 queues. We compared the results of these project compiling, such like the max clock frequency, max throughput and the utilization of each project.

TABLE I. SYNTHESIS REPORT

No.	Max Frequncy	Max throughput	Slice Registers Utilization
1 queue	157.035MHz	10G	2%(11,129/407,600)
2 queues	153.667MHz	19G	3%(13,252/407,600)
4 queues	145.328MHz	37G	3%(13,252/407,600)
No.	Slice LUTs		Occupied Slices Utilization
	Logic Utilization	Memory Utilization	
1 queue	4%(8,497/203,800)	1%(384/64,000)	9%(4,653/50,950)
2 queues	4%(9,879/203,800)	1%(392/64,000)	10%(5,289/50,950)
4 queues	4%(9,880/203,800)	1%(392/64,000)	10%(5,486/50,950)

B. *REPLAY PRECISION TEST*

1) Test environment

In this test, we need two host to setup our add-on board and one Smartbits packet generator.

HOST1: add **timestamp1** to the captured packets, so that we can record the characteristic of captured traffic. Then the packets were replayed to host2.

HOST2: add **timestamp2** to the replayed packets, so that we can compared the replay precision to the captured traffic.

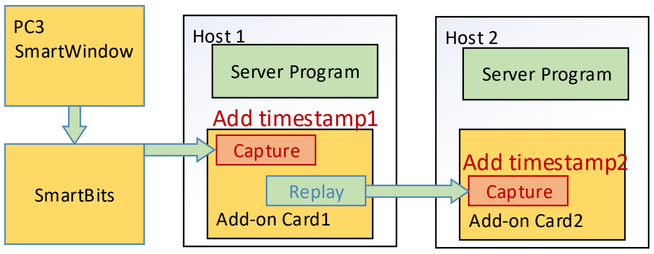

Fig. 6. The test framework of replay time precision

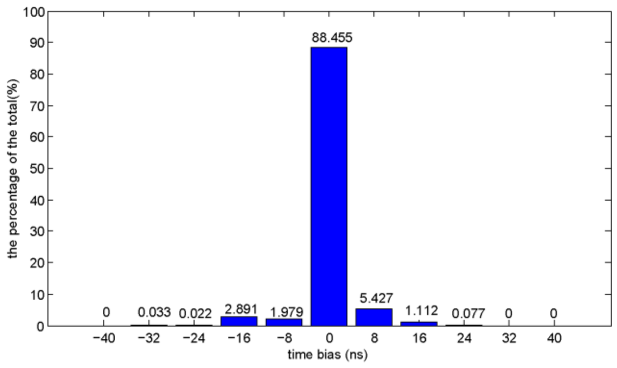

Fig. 7. The bias time distribution of all packets

2) Test result

We set packet generate rate to 1Gbps, and system running clock is 125MHz. We teste packets of 60, 200, 400, 600, 800, 1000, 1200, 1400 and 1480bytes length for our system. We define the **Δtimestamps(time bias)** as the precision of our replay system.

$$\Delta\text{timestamp} = \Delta\text{timestamp1} - \Delta\text{timestamp2 (ns)} \quad (1)$$

We find that 88.455% of the whole test packets replayed without any deviation, and only 0.033% of the whole test packets diverge the max Δtimestamp 32ns, as presented in *Fig. 7.*

Different length of packets test may generate a different deviation distribution, we show the 60bytes packets and 1480bytes packets test result in Fig. 8.

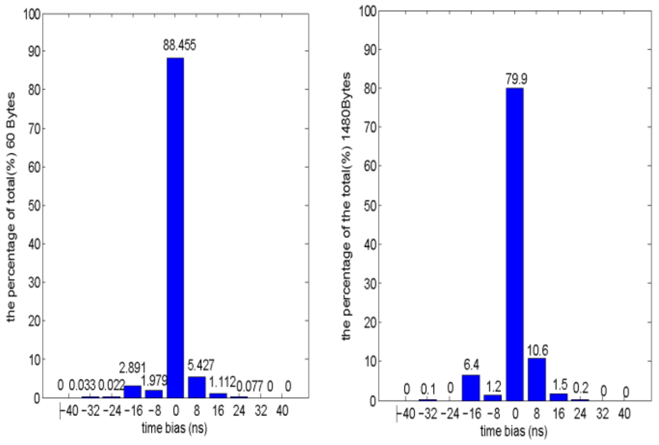

Fig. 8. The time bias distribution of 60Bytes and 1480Bytes packets

IV. DEMONSTRATION OVERVIEW

We can demonstrate the core logic in FPGA on our own designed add-on card right now. We will show a FPGA running circumstance for packets capturing, including all the processing function in FPGA user data path, for example the function of add-time-stamp, hash and arbiter.

V. CONCLUSION

In this paper, we realize a traffic capture and replay system based on The Add-on Card, interpret the detailed logic design of UDP pipeline on FPGA and then show the performance results of this system. In TC, we attached every packet with a 64 bit unique timestamp with an accuracy of 8 ns, and allocate incoming traffic to 4 CPU queues in line rate. And in TR, we send packets from 4 queues that stored by TC and replay to network from a single Ethernet interface without packet loss.

REFERENCES

[1] Chu W, Guan X, Gao L, et al. Model-based real-time volume control for interactive network traffic replay[C]. //Network Operations and Management Symposium (NOMS), 2012 IEEE. IEEE, 2012:163 - 170.

[2] L.Hu, J.Kang K.Zhao, et al. Intrusion detection systems [J]. Journal of Jilin University Information Science Edition, 2002, 46-47.

[3] TCPDUMP/LIBPCAP public repository[OL].http://www.tcpdump.org.

[4] Naous J, Gibb G, Bolouki S, et al. NetFPGA: reusable router architecture for experimental research[C].Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow. ACM, 2008: 1-7

[5] NetFPGA[OL].https://github.com/NetFPGA/netfpga/wiki/Guide.