NETFLOW PROBE INTENDED FOR HIGH-SPEED NETWORKS

Martin Žádník

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic
email: xzadni00@stud.fit.vutbr.cz

Tomáš Pečenka†, *Jan Kořenek*

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic
email: {pecenka, korenek}@fit.vutbr.cz

ABSTRACT

With growing speed of communication over the Internet there is a need for a reliable monitoring devices which are able to provide information about spectrum of traffic mix, attacks, applications, etc. This paper proposes architecture of network flow monitoring adapter based on hardware platform COMBO6. With use of field programmable gate arrays (FPGA) placed on these cards it is possible to monitor flows in high-speed environment. Component parts of the proposed architecture and implementation platform are described. Several different models have been created to analyze and prove important characteristics of the architecture and results are derived. The probe is able to monitor 1 million simultaneous flows on an 2Gbps network link.

1. INTRODUCTION

Providing accurate data about network traffic is important for managing network, bandwidth provisioning, detecting DoS attacks, billing and accounting. Although some active devices have monitoring abilities it is usually only a portion of the traffic about what they are able to give appropriate data. E.g. during a traffic peak or an attack the device is overwhelmed and unable to monitor whole bandwidth. That significantly decreases value of statistical data it exports. Monitoring abilities also differ from one device to another because a lot of methods have been introduced during last few decades. Each usually focuses on special type of problem. For example monitoring based on simple counters is suitable when network operator wants to know the utilization of the device but it is absolutely improper for billing of users or payload checking. Better solution is provided by devices where operators can specify rules for traffic they want to analyze. Unfortunately then it is not possible to reconstruct the whole traffic mix. Monitoring systems can also be optimized on certain type of flows (e.g. TCP flows [1]). But usually traffic mix consists of more than one type of flows.

NetFlow (general flow monitoring), first implemented in Cisco routers, is the most widely used measurement solution today [2]. Number of bytes, packets, flag fields, time of the flow are measured according to packets header. Keeping state for every flow allows to tell with whom, how long, at what intervals, with what protocol and port how much data was transfered [2]. Stand alone, dedicated device for such type of monitoring has several benefits: no need to change routers that do not support NetFlow, high speed of data processing, large flow cache, various enhancements to protect itself against malicious traffic (sideway filters [3], heuristics [4], etc.).

Nowadays FPGAs offer high performance in means of computational power, flexibility, maximal external memory bandwidth utilization and ratio of cost/performance. All these favorable features allow to optimize architecture for a specific application.

In this paper novel architecture of NetFlow monitoring probe is proposed and evaluated. Its implementation on the state-of-art hardware platform offers simultaneous processing of up to $1,000,000$ flows at speed of 2Gbps. It can also benefit from reprogrammable characteristic of FPGA so various enhancements are possible while the basic architecture remains the same.

2. ARCHITECTURE OVERVIEW

The data and control flow diagram is shown on Figure 1. The architecture consists of six main units which will be synthesized in FPGAs. Function of each unit is briefly described following paragraphs.

Incoming packets from network interface are entering the *Input Buffer* (IBUF) block. For each incoming packet the CRC is checked and only those ones with correct CRC pass through. Every packet is assigned a timestamp and saved into the internal IBUF memory, which serves as short-time storage for incoming packets.

*Aknowledgement to CESNET z. s. p. o. project No. 34602/2004 "Programmable hardware"

†Aknowledgement to project GA102/05/H050 - "Integrated approach to education of PhD students in the area of parallel and distributed systems".

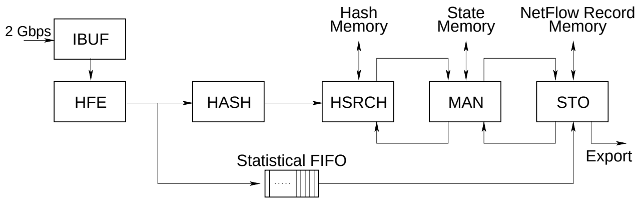

Fig. 1. Design flow diagram

Packets with assigned timestamp are then processed by *Header Field Extractor* (HFE). It is a processor based on RISC architecture controlled by specific instruction set intended for analyzing of input packets. It reads packet from the Input Buffer, analyzes control information in its headers, extracts required fields from IP and TCP/UDP headers and assemble the unique key which designates each flow. The key consists of IP source address, IP destination address, source port, destination port, transport layer protocol, type of service (ToS). After processing each datagram HFE is also able to provide required record for *Statistical FIFO*. Possible slow performance of HFE can be compensated by multiple instantiations of this unit.

The hash algorithms are made upon the key entering into the *Hash Unit* (HU). As result HU provides two types of hash values. The first one designates addresses where Hash Search (the next unit) should look up for entries. The second one determines whether addressed entries belong to the key (thus reducing probability of undetected collision). In addition, hash function (CRC with optional initialization of registers) has to be modifiable on demand of the user to protect the device before being tricked.

The *Hash Search* (HSRCH) lookups for convenient entry in its memory according to generated hash value (fixed number of additional lookups increase probability of finding empty entry). This operation can end up with several different results: no entry for the key is present and/or no empty entry is available, entry has been already created. Result and additional data are transferred to the Manager. It is also necessary to remove expired entries according to requests from Manager and acknowledge it back to the Manager. Look-up technique allows to remove entries easily, has low rate of false-positive and it is possible to store additional information in the entry.

The *Manager* (MAN) is responsible for keeping states of active and inactive flows. For this purpose bidirectional bounded list is proposed. New or updated flows are rebounded to the top of the list so the oldest ones remain at the bottom. Then it is easy to identify inactive flows and send delete request to HSRCH. Its other duty is to copy information coming from Hash Search to Storage unit and vice versa. Since Storage can also generate delete requests it is up to Manager to guarantee no duplicate requests to Hash Search.

Statistical data is held in *Storage* memory. The unit reads data from Statistical FIFO and performs different operation according to commands provided by Manager. The unit is responsible for checking if records are not held for too long (starting timestamp of the flow is compared with actual timestamp while flow statistical data is updated). If so, delete request is sent to Manager.

To sum up there is a sparsely occupied memory that provides large address space in Hash Search. Therefore the probability of finding empty entry for new flows after several lookups is very high. Another advantage is that the entry is not wide in comparison to whole record we want to hold for one flow so the memory can have smaller capacity and is better utilized than it would be if the whole record was stored there. The advantage of bidirectional sorting list implemented in Manager is precise export of inactive flows without need of large lookup in the memory. Unfortunately to implement it, multiple accesses to memory in short time are required so it is necessary to implement it in SSRAM memories.

Last block is intended to keep up to $1,000,000$ flow records (64 byte each). Only DRAM memory offers such a large capacity.

3. IMPLEMENTATION PLATFORM

The proposed architecture is suitable for COMBO6 cards developed by Liberouter project [5]. COMBO6 is a PCI card, which can be used in various applications. It consists of Xilinx Virtex–II XC2V3000 FPGA, 2MB Ternary CAM, 256MB DRAM and three 2MB SSRAM (see [6] for details). Various add-on cards can be used with COMBO6 card. For purposes of NetFlow project new interface card with two 10GE interfaces, Xilinx Virtex–II XC2VP20 FPGA and four SSRAMs (4MB each) is developed nowadays. COMBO6-PTM card could also support NetFlow probe as a source of precise timestamp [6].

With such hardware resources the architecture allows to monitor 2Gbps link in full rate and 10Gbps in sampled rate with simultaneous processing of up to $1,000,000$ flows.

4. EVALUATING THE ARCHITECTURE

The architecture for monitoring has to be very accurate, reliable and with required performance. Following chapter proves that demands on low error rate can be reached with properly chosen sizes of external memories. So the design on FPGA remains nearly the same. Throughput determines the link speed the device can operate on as well as its behavior in extreme situation. And so the next chapter explores what throughput can be reached when units communicate with each other supposing the traffic mix is known. In our case we consider the device is working under normal condition and also under attacks. The advantage of modeling the system is that the network operator knows behavior of the monitoring system in most serious cases. It is also a valuable information for designers because they can find out what causes the bottleneck of their design and they can focus on its optimization. Output rate can be also derived which is useful for higher-level programmers of drivers (throughput of PCI, size of buffer).

4.1. Hash

Let us suppose that the Hash Search is able to distinguish whether the flow belongs to certain entry or not. V is the ratio of occupied entries to number of all entries. Then the probability that Hash Search finds no empty item for new flow is

$$P(failure) = V, \tag{1}$$

which is too high. That is why additional lookups have to be done. Then it holds that after n lookups, probability of no available item is geometric sum:

$$P(failure) = V * \frac{1 - V^n}{1 - V} \tag{2}$$

Note the exponential dependency between number of lookups and the probability of failing to find empty entry (Figure 2). Four curves shows the dependency on the occupation of the memory. The designer can decide what size of memory to choose to fit in desired probability. The probability should be around $1 * 10^{-4}$ to guarantee maximal limit rate for forced export. That is when statistical record is removed without meeting inactive or active flow timeout.

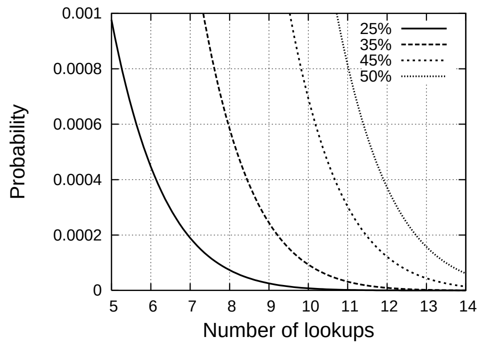

Fig. 2. Graph showing failure of finding empty entry after n lookups

As supposed above, the Hash Search is able to distinguish whether the flow belongs to certain entry or not according to another hash number which is stored in every entry point. That is why it is not crucial when first hash map two different flows to one entry.

Let us also suppose that we have a complete set of F possible flows. Every packet is assigned to one of B buckets (by hash function of length h), where $B < F$. However, packets of the same flow are always assigned to the same bucket. Collision is a situation when at least one of the buckets contains packets from at least two different flows. We are interested in number of such undetectable collisions per second:

$$B = 2^h \tag{3a}$$

$$P(collision) = \frac{k * S}{F} = \frac{k * \frac{F}{B}}{F} = \frac{k}{B} \tag{3b}$$

$$\frac{collision}{second} = P(collision) * x * y \tag{3c}$$

where S is number of flows belonging to one bucket, k is number of used items, x number of new flows per second and y number of lookups per one incoming packet. According to equations 3 the number of undetectable collisions per second is exponentially dependent on the hash length. For example:

$$\frac{col}{s} = \frac{2^{20}}{2^{63}} * 2^{21} * 14 = 3.3 * 10^{-6} \tag{4}$$

4.2. Mass Service System Model

The M/D/1 Kendall mass service system model is used for the analytic system model of packet processing in critical part of the design. We are interested in throughput of the design during different traffic mix. The length of each queue is monitored to find out whether belonging unit is able to process all requests. There is a simplified view of the model based on Petri Net on Figure 3.

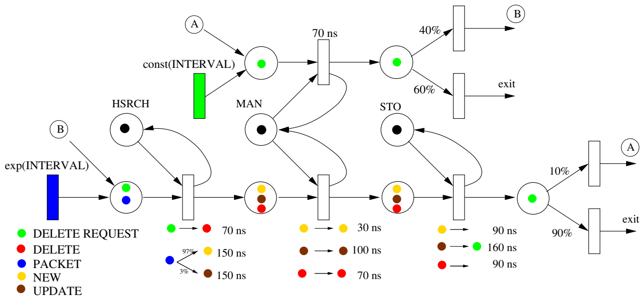

Fig. 3. Model of system based on Petri Net and token with attributes

It is very hard to create mathematical model of such a large system and verify it. Other techniques can reach nearly as good results as theoretical solution even in shorter time. Our model was implemented according to Figure 3 in C++/SIMLIB [7]. Integral part of model are memories because they determine which command is generated out of the unit. Time of service differs from one unit to another and also from one command to another.

Experiment setup: Memory tables are filled up to 50 percent with subsequent packets each creating new flow much like an attack. Then simulation starts and runs for 1s with an attack (interval between packets is constant, 98 percent of packets create new flow).

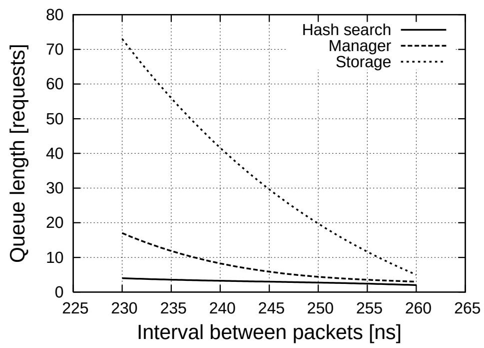

Fig. 4. Maximal length of queues

The mass service model evaluates throughput of the design according to maximal length (maximal length of the queue does not have to exceed 32 requests) of queue prior to every unit 4. Please note that IBUF and HFE units are not present in the model. It is because they can be instantiated multiple times in parallel thus increasing their throughput.

Results of simulations confirm that throughput of the model is suitable for two gigabit links. Proposed architecture is able to process 4 million 64-byte packets per second (Figure 4) when every packet creates new flow (the worst case).

4.3. Hardware resources utilisation

New add-on card for NetFlow adapter is designed now. Let us have a closer look how its hardware resources could be utilized in the most efficient way. Four SSRAMs (4MB each) memory modules are convenient as Hash Search memory of entries. From graph on Figure 2 can be seen that utilization of this memory does not have to exceed fifty percent when after fourteen lookups the probability of no available empty entry should be reasonably low (0.0001). Given goal is to monitor one million simultaneous flow which requires 2^{21} of entries. The entry is 64 bit wide in respect to available memory. Then the first hash is 21 bits long and the second one 42 bits long (the rest of the entry is occupied by pointer to Manager and/or Storage memory). The number of undetected errors is then $3.3 * 10^{-6}$ per second (1 error per 84 hour) Equation 4.

5. CONCLUSION

In the paper the architecture of high performance flow monitoring adapter was presented. The architecture was evaluated to find out its limits in means of throughput and error rate. Utilizing COMBO6 cards the proposed architecture allows to monitor two gigabit links with 1 million simultaneous flows. This design can be enhanced to monitor 10GE network link by employing input sampling unit (rate $1/5$). Thanks to FPGA technology various heuristics (e.g. sample and hold [8], renormalization, sideway filters [9], $\ldots$) can be added to increase power and monitoring abilities. Our future work concentrates on more simulations and tests as well as on implementing the architecture in VHDL.

6. REFERENCES

[1] D. Schuehler and J. Lockwood, "A modular system for FPGA-based TCP flow processing in high-speed networks," in *14th International Conference on Field Programmable Logic and Applications (FPL)*, Antwerp, Belgium, Aug. 2004, pp. 301–310.

[2] "Cisco," 2005. [Online]. Available: www.cisco.com

[3] N. Duffield and C. Lund, "Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM Press, 2003, pp. 179–191.

[4] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Trans. Comput. Syst.*, vol. 21, no. 3, pp. 270–313, 2003.

[5] "Liberouter," 2005. [Online]. Available: www.liberouter.org

[6] J. Novotný, O. Fučík, and R. Kokotek, "Schematics and PCB of COMBO6 Card," CESNET, Tech. Rep., 2002. [Online]. Available: http://www.cesnet.cz/doc/techzpravy/2002/-combo6/combo6.pdf

[7] P. Peringer, "SIMLIB/C++," 2004. [Online]. Available: http://www.fit.vutbr.cz/ peringer/SIMLIB/index.html.en

[8] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 245–256, 2004.

[9] T. Košnár, "Notes to flow-based traffic analysis system design," CESNET, Tech. Rep., 2004.