FPGA Accelerated Change-Point Detection Method for 100 Gb/s Networks

Tomáš Čejka[1], Lukáš Kekely[1], Pavel Benáček[2], Rudolf B. Blažek[2], and Hana Kubátová[2]

[1] CESNET a. l. e.
Zikova 4, Prague, CZ
cejkat,kekely@cesnet.cz
[2] CTU in Prague – FIT
Thakurova 9, Prague, CZ
benacekp,rblazek,hana.kubatova@fit.cvut.cz

Abstract. The aim of this paper is a hardware realization of a statistical anomaly detection method as a part of high-speed monitoring probe for computer networks. The sequential Non-Parametric Cumulative Sum (NP-CUSUM) procedure is the detection method of our choice and we use an FPGA based accelerator card as the target platform. For rapid detection algorithm development, a high-level synthesis (HLS) approach is applied. Furthermore, we combine HLS with the usage of Software Defined Monitoring (SDM) framework on the monitoring probe, which enables easy deployment of various hardware-accelerated monitoring applications into high-speed networks. Our implementation of NP-CUSUM algorithm serves as hardware plug-in for SDM and realizes the detection of network attacks and anomalies directly in FPGA. Additionally, the parallel nature of the FPGA technology allows us to realize multiple different detections simultaneously without any losses in throughput. Our experimental results show the feasibility of HLS and SDM combination for effective realization of traffic analysis and anomaly detection in networks with speeds up to 100 Gb/s.

1 Introduction

Computer networks are getting larger and faster, and hence the volume of data captured by network monitoring systems increases. Therefore, there is a need to analyze more data for detection of network attacks and traffic anomalies. This paper deals with real-time detection of attacks suitable for high-speed computer networks thanks to the direct deployment of detection methods in hardware monitoring probe.

Today, monitoring systems usually consist of several probes that capture and preprocess huge amounts of network traffic at wire speed, and one or more collector servers that collect and store network traffic information from these probes. Analysis of network data is traditionally also realized at the collectors. In this paper, we propose a different approach, where anomaly detection is shifted directly into the monitoring probes. The aim of this approach is to enable real-time analysis even in very large networks with speeds up to 100 Gb/s per Ethernet port and to reduce the latency of anomaly detections.

It is virtually impossible to process all network data from the 100 Gb/s link in software using only commodity hardware. The main limitations lay in insufficient bandwidth of communication paths between the network interface card and the software components [1] and in limited performance of the processors. Therefore, hardware acceleration must be used for high-speed networks in order to avoid transferring and processing of all the data in the software.

In this paper, we utilize a special network interface card mounted with FPGA chip for hardware acceleration of network traffic processing as a basis for our high-speed probe. The FPGA on the card allows us to realize more advanced data processing features (e.g. anomaly detection methods that use packet level statistics) directly on the card, thus reducing the data load for the software. To demonstrate this approach, we concentrate on a real-time sequential Change-Point Detection (CPD) method that is designed to minimize the average detection delay (ADD) for a prescribed false alarm rate (FAR) [2,3].

As the basis for the FPGA firmware, we use Software Defined Monitoring (SDM). SDM is a novel monitoring approach proposed in [4], that can be used as a framework for hardware acceleration of various monitoring methods. SDM combines hardware and software modules into a tightly bound co-design that is able to address challenges of monitoring from data link to application layer of the ISO/OSI model in modern network environments at the speeds up to 100 Gb/s.

The main contribution of this paper is the evaluation of a statistical real-time detection methods implemented in hardware. The detection methods are extensions of a hardware accelerated monitoring probe designed for 40 Gb/s and 100 Gb/s Ethernet lines. The resulting device is able to analyze unsampled high-speed network traffic without loss.

The rest of this paper is organized in the following way. Introduction to the implemented sequential non-parametric change-point detection method (NP-CUSUM) can be found in Sec. 2. The used SDM concept is briefly described in Sec. 3. Sec. 4 describes created hardware implementation of detection method. Evaluation of the developed system and the achieved results are presented in Sec. 5. Related work and main differences between existing projects and our implementation are presented in Sec. 6. Sec. 7 summarizes the results presented in this paper and outlines our future work.

2 Change-Point Detection

Network attacks, intrusions, or anomalies appear usually at unpredictable points in time. The start of an attack is mostly observable as a change of some statistical properties of the network traffic or its specific part. Therefore, methods based on sequential Change-Point Detection theory are suitable for intrusion detection. CPD methods detect the point in time where the distribution of some perpetually observed variables changes. In network security settings, these variables correspond to some relevant, directly observed or calculated network traffic characteristics. The main problem of such approach is the lack of precise knowledge about the statistical distributions of these traffic characteristics. Ideally, the distributions should be known both, before and after the distribution change that corresponds to the anomaly or attack. Therefore, we use a non-parametric CPD method NP-CUSUM that was developed in [2,3] and that does not require precise knowledge about these statistical distributions.

NP-CUSUM is inspired by Page's CUSUM algorithm that is proven to be optimal for detection of a change in the mean (expectation) when the distributions of the observed random variables are known before and after the change [5]. The typical optimality criterion in CPD is to minimize the average detection delay (ADD) among all algorithms whose average false alert rate (FAR) is below a prescribed low level. Page's CUSUM procedure, which is based on the log-likelihood ratio, can for i.i.d. (independent and identically distributed) random variables X_n be rewritten [5] as:

$$U_n = \max\left\{0, U_{n-1} + \log\frac{p_1(X_n)}{p_0(X_n)}\right\}, \quad U_0 = 0, \tag{1}$$

Where p_0 and p_1 are the densities of X_n before and after the change, respectively.

The formulation in (1) is the inspiration for the NP-CUSUM method procedure [2,3]. The procedure is applicable to non–i.i.d. data with unknown distributions (i.e. the method is non-parametric). First, the Page's CUSUM procedure was generalized as $S_n = \max\{0, S_{n-1} + f(X_n)\}$ with some function f. Changes in the mean value of X_n can be detected using sequential statistic:

$$S_n = \max\{0, S_{n-1} + X_n - \hat{\mu} - \varepsilon\hat{\theta}\}, \quad S_0 = 0, \tag{2}$$

Where $\hat{\mu}$ is an estimate of the mean of X_n before the attack, $\hat{\theta}$ is an estimate of the mean after the attack started, and ε is a tuning parameter for optimization. It has been shown in [3] that with optimal value of ε the NP-CUSUM procedure (2) is asymptotically optimal as FAR decreases. That is, for small prescribed rate of false alarms, other procedures will have longer detection delays. In fact, the delays can theoretically be exponentially worse [3].

As the input of the NP-CUSUM algorithm, we can use various features X_n of the observed network traffic. To basically evaluate our hardware implementation of the method, we have chosen for X_n the ratio of SYN and FIN packets of the Transmission Control Protocol (TCP) in a short time window [6]. During "normal" operation of the network, each connection is opened using two SYN packets, and closed using two FIN packets (one in each direction). Therefore, we expect the ratio of SYN and FIN packets to be on average close to 1 or at least constant. Sudden and consistent change of the ratio is suspicious and can be caused by some sort of attacks (e.g. SYN or FIN packet flood) [6].

To demonstrate the scalability and power of our hardware implementation using SDM, we raise the number of observed statistics and add some more NP-CUSUM blocks in parallel. The added statistics utilize information about ICMP and RST TCP packets. All measured values are used in form of ratios in order to avoid the dependency on trends and traffic volumes that could increase the number of false alerts. Finally, thanks to parallelism, observation of multiple statistics simultaneously does not negatively affect the processing throughput.

3 Software Defined Monitoring System

Software Defined Monitoring (presented in [4,7]) forms a basis for our hardware implementation of detection methods in a monitoring probe. In this section we briefly describe the main architecture of the SDM system and the changes needed to accommodate the implementation of NP-CUSUM monitoring system.

An SDM system consists of two main parts: firmware for the FPGA on hardware accelerator and software for general processors. The hardware and software components are connected via a PCI-Express bus. Both parts are tightly coupled together to allow precise software control of hardware processing. The software part of the SDM system consists of monitoring applications and a controller. The monitoring applications can perform advanced monitoring tasks (such as analysis of application protocols) or also export information (alerts) to the collector. The controller manages the hardware module by dynamically removing and inserting processing rules into its memory (see Fig. 1). The instructions contained in the rules tell the hardware what actions to perform for each input packet with some characteristics. These rules are defined by the monitoring *Applications*, which inserts them to the *Hardware* via the *Controller*.

Due to aforementioned facts, the monitoring application can not only use data coming from the hardware, but it can also manage the details of hardware processing of network traffic as well. The offloading of traffic processing into the hardware saves both, the bandwidth of communication interface (PCIe) and the CPU processing time. The hardware module can pass information to the software in the form packet metadata from a single packet, or as aggregated records computed from multiple subsequent packets with common features (such as NetFlow [8] aggregation). Whole received packets or their parts can be also sent to the software for further (deeper) analysis. Graphical representation of the SDM concept is shown in Fig. 1.

Processing of an incoming network packet in the SDM hardware starts with the extraction of its protocol headers. The extracted data are used to search adequate rule in memory that specifies the desired processing possibly supplemented by address of a record. The selected rule and metadata for each given packet are then passed to the SDM *Update* block, which is the heart of the SDM concept making that idea strong. This block contains a routing table that is used to forward the incoming processing request to the appropriate update (instruction) blocks, for execution. Each of these instruction blocks can perform a specific update operation (realize a specific aggregation type) on the record. Each update operation is delimited by two memory operations: reading the stored record values, and writing back the updated values. Also, new types of updates (aggregations) can be specified, simply by implementing the new instruction block

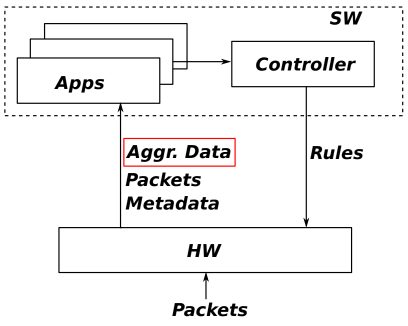

Fig. 1. Software Defined Monitoring (SDM) abstract architecture

and plugging it into the existing *Update* block infrastructure. A special type of processing action is an export into the software of the processed packet data, metadata, or stored values from a selected record, optionally followed by clearing of that record. Records can be exported when some special condition is met or in periodical manner.

4 Implementation

4.1 The CPD hardware block

Our hardware implementation of CPD method is realized as hardware plug-in for the SDM system. More precisely, it is available as a new instruction block for the SDM *Update* module that is described in the previous section. The SDM design supports access to arbitrary data records stored in memory for instruction blocks. Although, the available data size of a record is limited due to memory block size that can be read or written on each clock cycle – the block size is equal to 288 b. Usage of bigger data records than 288 b would cause unwanted latency increase and lower throughput of the whole monitoring hardware.

One CPD instruction block uses available space in memory to store: previous historical value, 2 parameters of the NP-CUSUM algorithm, and 1 threshold value that is used for alerting purposes. Memory should also contain counters with observed features such as the number of SYN or FIN packets, and the packet counter that starts the ratio and NP-CUSUM computation. The data stored in memory is accessible from software and therefore all of the thresholds and parameters can be changed on the fly.

The source code of the instruction block allows us to specify the data type size of all values stored in memory. The choice of data type sizes implies the number of hardware blocks that can work in parallel in the same clock cycle with the same memory block. However, the decrease of data type size lowers the value precision and data ranges. The NP-CUSUM parameters, the previous historical value and the threshold are represented as 16 b decimal numbers. The counters are set to 8 b. For one block that analyzes SYN/FIN ratio, the implementation works with 88 b of memory for one record in total. Configuration with 4 NP-CUSUM blocks uses 5 counters (SYN, FIN, RST, ICMP, packet counter) and 4 sets of fixed-point values. In total, 4 NP-CUSUM blocks would use 296 b of memory. Therefore the size of decimal number data type was shortened to 15 b and the total used memory size was decreased to available 280 b.

We use a high-level synthesis (HLS) approach [9], to implement the CPD method from Sec. 2 for the FPGA as an instruction block inside the SDM system. The structure of the implemented block is shown at Fig. 2. The main advantage of using HLS approach is faster implementation of new hardware accelerated monitoring and detection methods with minimal loss of efficiency in comparison to traditional coding of FPGA firmware using Hardware Description Languages (HDL) such as VHDL or Verilog. Following the requirements for the SDM instruction block interfaces and general behavior, we have developed the CPD hardware block in the C++ language.

Implementation of the CPD hardware block brings a several issues to solve. The most important one is the choice of decimal numbers representation. We try two of the standard approaches: fixed-point and floating-point representation. The main advantage of the floating-point approach is the ability to represent a greater range of values. But on the other hand, hardware realization of floating-point arithmetic is very complicated and considerably slower. Therefore, the usage of fixed-point arithmetic can be favored by better performance and lower resource usage of the instruction block.

From the HLS point of view, the most important parameter for our design goals is the achievable Initiation Interval (II). This parameter represents the number of clock cycles needed for initialization of a new request in the instruction block. Ideally, we require the II to be equal to one so that a new request can be accepted in each clock cycle and the instruction block is able to achieve full throughput. During our experiments, we have discovered that the effect of decimal numbers representation on the II is following: floating-point version of the instruction block has II of 11 clock cycles whereas the fixed-point version has II of 1.

Another very important performance-related parameter of our implementation is latency. It is required to be as small as possible because high latency can lead to delays between repeated processing of the same instruction caused by the fact that records in the memory need to be locked in order to achieve atomic processing. In the end, our experimental timing and performance results indicate that the created implementation is able to handle network traffic at 100 Gb/s Ethernet line. More detailed results regarding our synthesis and FPGA requirements are discussed in Sec. 5.

Apart from CPD instruction block creation, another important part of the implementation is connection of the new instruction block to the existing SDM *Update* block. Thanks to the by design extensibility of SDM *Update* block, this task is simple and straightforward. All that needs to be done is to wrap the translated HLS implementation of the new block in a VHDL envelope that is responsible for adapting the behavior of all predefined interface signals. The

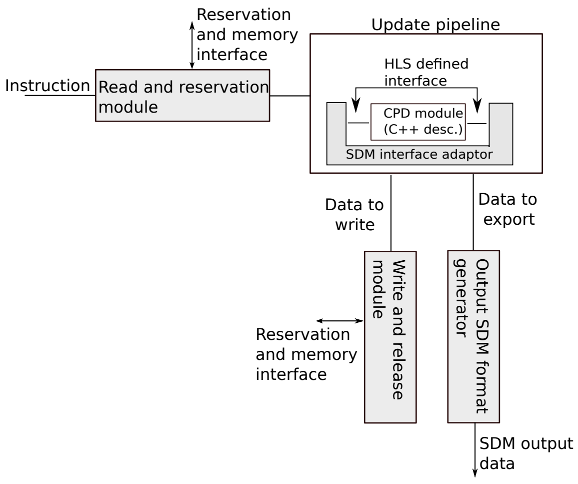

Fig. 2. Implementation of the CPD Instruction

wrapping process is depicted in Fig. 2. The gray blocks are parts of the SDM designated for connecting of a new instruction blocks. The SDM can thus be viewed as some kind of a framework that brings the possibility to create new hardware modules for rapid network monitoring acceleration.

To finish the implementation of the Change-Point Detection method in the SDM system, a software monitoring application needs to be created. The application communicates with an SDM Controller daemon to manage the detection details in hardware module (see Fig. 1) and also receives detected alerts. The main task of the monitoring application is to control the detection process and present its results to human operators.

5 Evaluation

Correct functionality of the created implementation of the CPD block was verified using referential software application. The referential application is written in the plain C language and is not meant to be highly optimized for the HLS. Its main purpose is only to validate the functionality of the hardware implementation. In addition, the software application is extended and serves as the base for the measuring and detection application [10] that can be used in slower networks or for estimation of configurable parameters for the CPD block.

We have implemented the hardware-based prototype of the NP-CUSUM detection method as an instruction block for the SDM Update block in an SDM monitoring probe. The prototype is developed for the network interface card with a 100 Gb/s Ethernet port and a Virtex-7 H580T FPGA, which is the main core for the implemented detection functionality.

A detailed list of all FPGA resources needed for the implementation of one CPD instruction block, which observes one feature, is shown in Tab. 1. In the table there are also results for other constellations of the CPD blocks that contain more computational blocks with 1, 2, or 4 instances of the NP-CUSUM algorithm and observe more features in parallel. The total number of available resources on used chip is 725 600 Flip-Flops (FF) and 362 800 Look-up tables (LUT). The number of utilized LUTs and FFs for CPD instruction block itself, therefore, accounts only for less than 1% of the available FPGA resources.

Table 1. FPGA resources used for the CPD instruction block in different configurations.

Name	1 block		2 blocks		4 blocks	
	FF	LUTs	FF	LUTs	FF	LUTs
Expression	0	458	0	496	0	479
Instance	280	252	560	504	560	504
Multiplexer	-	1842	-	1868	-	2130
Register	2253	-	2377	-	2593	-
ShiftMemory	0	806	0	816	0	814
Total	2533	3358	2937	3684	3178	3982

Performance results for the CPD instruction blocks are shown in Tab. 2 and Tab. 3, whereas Tab. 3 shows detailed information about the fixed-point implementation. An Initiation Interval is required to be equal to one in order to support processing of 100 Gb/s network traffic at full wire-speed (see Sec 3). This requirement is not satisfied only by the floating-point implementation. Vivado HLS version 2013.2 was used for high-level C to VHDL synthesis. Xilinx ISE version 14.7 with enabled synthesis optimization was used for VHDL to FPGA netlist synthesis. Enabling the optimization such as register duplication leads to a higher clock frequency achieved for the final implementation and also to a higher resources consumption. The tables illustrate that after the optimization all performance requirements from Sec. 3 have been met by the fixed-point implementation.

Table 2. Comparison of timing results for the synthesized CPD instruction blocks.

Parameter	**Reached Fixed-point**	**Reached Floating-point**	**Required**
Clock period	4.08 ns	16.48 ns	5 ns
Frequency	245 MHz	60.679 MHz	200 MHz
Latency	12	11	-
Initiation Interval	1	12	1
Bus Width	512 b	512 b	512 b
Achieved Throughput	125 Gb/s	2.5 Gb/s	100 Gb/s

Table 3. Performance results for the CPD instruction blocks in different configurations.

Parameter	**Reached 1 block**	**Reached 2 blocks**	**Reached 4 blocks**	**Required**
Clock period	4.08 ns	4.20 ns	4.20 ns	5 ns
Frequency	245 MHz	238 MHz	238 MHz	200 MHz
Latency	12	12	12	-
Initiation Interval	1	1	1	1
Bus Width	512 b	512 b	512 b	512 b
Achieved Throughput	125 Gb/s	121 Gb/s	121 Gb/s	100 Gb/s

Finally, Tab. 4 shows the total number of FPGA resources required for the whole synthesized SDM system with one CPD hardware plug-in. The table shows that about 87% of the Virtex-7 H580T resources are still available. Therefore, it is feasible to include several CPD hardware plug-ins in the SDM system for parallel detection of various anomalies without significant latency increase nor throughput loss.

Table 4. FPGA resources of the SDM system with one CPD hardware plug-in (FPGA xc7vh580thcg1155-2).

Resource Name	Used Resources [-]	Utilization Percentage
LUTs	47731	13 %
Registers	21089	2 %
BRAMS	107	11 %

6 Related Work

We present a brief overview of related work with regard to the differences of our work. This section can be divided into two main domains. The first domain is related to the hardware accelerated detectors and the second domain is related to the detection methods. From the hardware point of view, there are two interesting projects somehow similar to our – Gorilla and Snabb Switch.

The Gorilla project [11] is the closest comparable solutions that we found. Gorilla is a methodology for generating FPGA-based solutions especially well-suited for data parallel applications. The main goal of Gorilla is the same as our goal in SDM Update – to make the hardware design process easier and faster. Our solution is however specially designed for the stateful processing of network packet data. Furthermore, SDM is able to work with L2–L7 layers of ISO/OSI model. In addition, the resource consumption of Gorilla is higher than our solution.

The Snabb Switch project [12] shows different approach of network packets processing. This approach uses modified drivers for faster transfer of network packets from the network interface card to computer's memory. Transferred data are then processed by network applications. There is also available a Snabb Lab with an accessible platform for measuring. This platform consists of the Supermicro motherboard with dual Xeon-E5 and 20x10 GbE (Intel 82599ES) network cards. This configuration allows to process network traffic at speed of 200 Gb/s. Massive usage of this platform is complicated due to large number of network cards. Our solution is able to process network traffic at speed of 100 Gb/s on one Ethernet line (2 ports allows to achieve 200 Gb/s). Our work is focused on full hardware acceleration of network traffic processing using the only one 100 Gb/s Ethernet port.

From the detection method point of view, there are various existing approaches of anomaly detection from many authors. Detection of SYN flood attacks have been studied and well described in many papers. However, this issue is currently still relevant because of increase of network traffic volumes. Detection based on NP-CUSUM is used in [13] by Wang et al., where the authors present their observation about SYN-FIN pairs in network traffic under normal condition: (1) there is a strong positive correlation between the SYN and RST packets; (2) the difference between the number of SYN and FIN packets is close to the number of RST packets. The authors bring experimental evaluation of flood detection using NP-CUSUM, however they mention a possible disadvantage of aggregated counting of packets that can be spoofed by emission of mixed packet types by attacker.

Siris et al. in [14] compare a straightforward adaptive threshold algorithm, which can bring satisfactory performance for attacks with high intensity, and algorithm base on cumulative sum (CUSUM). Adaptive threshold algorithm uses a difference from moving average value computed e.g. by EWMA algorithm. An alarm is signalized when measured value is higher then moving average in last k consecutive intervals. The CUSUM variant of detection algorithm is influenced by seasonality and trends of network traffic (weekly and daily variations, trends and time correlations). The authors propose to use some prediction method to remove non-stationary behavior before applying the CUSUM algorithm. However, because of time-consuming calculations with minor gains compared to simpler approaches, the authors used simpler approach based on application of CUSUM on difference between measured value and result of Exponential Weighted Moving Average (EWMA) [15] algorithm.

Smoothing of the data signal is important for minimizing the number of false alarms that can be caused by high peaks in data. Therefore, the data are usually preprocessed to avoid short-time deviations to detect long-time anomalies. There are various approaches to smooth the signal and the possible way is to exploit some prediction method such as Moving average, EWMA, Holt-Winters [16], or Box-Jenkins (ARIMA) [17] methods. However, dependency of an algorithm on historical and current measured values can be dangerous and can lead to overlooking of an attack. The issue of self-learning and self-adaptive approach is being studied in our current and future work, however, it is out of the scope of this paper.

Salem et al. presented the currently used methods of the network anomaly detection in [18]. The paper evaluates the usage of extended NP-CUSUM called Multi-chart NP-CUSUM, proposed by Tartakovsky et al. in [19], in combination with Count Min Sketch and Multi-Layer Reversible Sketch (sketching method is proposed eg. in [20]) for data aggregation and anomaly detection.

This paper is focused on the hardware implementation of the detection method, whereas other authors usually more or less rely on software processing of aggregated data. Our solution allows the detection method to be real-time and independent on overloaded software part of system.

7 Conclusions

In this paper we present implementation and evaluation of the CPD algorithm (NP-CUSUM) as hardware plug-in for the Software Defined Monitoring system. We achieve easy and rapid development of detection hardware blocks for the FPGA thanks to the usage of high-level synthesis. Also, creation of monitoring probe utilizing newly implemented detection method is very simple and straight forward thanks to the utilization of SDM as the platform for high-speed packet processing. Moreover, we show frequency and FPGA resource evaluation of the hardware implementation for the Virtex-7 H580T FPGA, which is large enough and fast enough to accommodate complex network processing.

Results presented in this paper show that our implementation of NP-CUSUM is capable of processing network traffic at the speed up to 100 Gb/s. The firmware of the whole monitoring probe consumes only 13 % of the available resources of the target FPGA and thus leaves space for several additional CPD (NP-CUSUM) hardware plug-ins that can be used for parallel detection of multiple kinds of network anomalies concurrently. In addition, other existing detection methods can potentially be easily implemented in the similar way – as hardware SDM plug-ins for detection of abrupt changes of network traffic characteristics. The limiting factor for deploying detection hardware plug-ins into a monitoring probe is the consumption of FPGA resources. Generally, detection methods with low data storage requirements can be fully implemented as a hardware plug-ins. Moreover, SDM allows creation of hardware-software co-design where only the most critical parts of the more complex detection algorithm can be accelerated. This partially hardware-accelerated approach can reduce the FPGA resource requirements of advanced detection methods with moderate performance loss.

Acknowledgment

This work is partially supported by the “CESNET Large Infrastructure” project no. LM2010005 funded by the Ministry of Education, Youth and Sports of the Czech Republic and the project TA03010561 funded by the Technology Agency of the Czech Republic.

References

1. Santiago del Rio, P.M., Rossi, D., Gringoli, F., Nava, L., Salgarelli, L., Aracil, J.: Wire-speed statistical classification of network traffic on commodity hardware. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference. IMC '12, New York, NY, USA, ACM (2012) 65–72
2. Blažek, R.B., Kim, H., Rozovskii, B., Tartakovsky, A.: A novel approach to detection of "denial–of–service" attacks via adaptive sequential and batch–sequential change–point detection methods. In: Proc. 2nd IEEE Workshop on Systems, Man, and Cybernetics, West Point, NY. (2001)
3. Tartakovsky, A.G., Rozovskii, B.L., Blažek, R., Kim, H.: A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. IEEE TRANSACTIONS ON SIGNAL PROCESSING **54**(9) (2006) 3372–3382
4. Kekely, L., Puš, V., Kořenek, J.: Software defined monitoring of application protocols. In: INFOCOM 2014. The 33rd Annual IEEE International Conference on Computer Communications. (2014) 1725–1733
5. Page, E.S.: Continuous inspection schemes. Biometrika **41**(1/2) (1954) 100–115
6. Wang, H., Zhang, D., Shin, K.: Detecting syn flooding attacks. In: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Volume 3. (2002) 1530–1539
7. Puš, V.: Monitoring of application protocols in 40/100gb networks. In: Campus Network Monitoring and Security Workshop, Prague, CZ, CESNET (2014)
8. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (2004)
9. Feist, T.: Vivado design suite. White Paper (2012)
10. Čejka, T.: Fast TCP Flood Detector. `http://ddd.fit.cvut.cz/prj/FTFD` (2014)
11. Lavasani, M., Dennison, L., Chiou, D.: Compiling high throughput network processors. In: Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays. FPGA '12, New York, NY, USA, ACM (2012) 87–96
12. Gorrie, L.: Snabb switch. `http://www.snabb.co` (2014)
13. Wang, H., Zhang, D., Shin, K.: Detecting syn flooding attacks. In: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Volume 3. (2002) 1530–1539
14. Siris, V.A., Papagalou, F.: Application of anomaly detection algorithms for detecting SYN flooding attacks. Computer communications **29**(9) (2006) 1433–1442
15. Ye, N., Borror, C., Zhang, Y.: Ewma techniques for computer intrusion detection through anomalous changes in event intensity. Quality and Reliability Engineering International **18**(6) (2002) 443–451
16. Brutlag, J.D.: Aberrant behavior detection in time series for network monitoring. In: LISA. (2000) 139–146
17. Box, G., Jenkins, G., Reinsel, G.: Time Series Analysis: Forecasting and Control. Wiley Series in Probability and Statistics. Wiley (2013)
18. Salem, O., Vaton, S., Gravey, A.: A scalable, efficient and informative approach for anomaly-based intrusion detection systems: theory and practice. International Journal of Network Management **20**(5) (2010) 271–293 00019.
19. Tartakovsky, A.G., Rozovskii, B.L., Blažek, R.B., Kim, H.: Detection of intrusions in information systems by sequential change-point methods. Statistical Methodology **3**(3) (2006) 252–293
20. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: methods, evaluation, and applications. In: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, ACM (2003) 234–247