Computing and Informatics, Vol. 37, 2018, 142–164, doi: 10.4149/cai_2018_1_142

ADAPTIVE AGGREGATION OF FLOW RECORDS

Adrián Pekár, Martin Chovanec

Institute of Computer Technology
Technical University of Košice, Slovakia
e-mail: {adrian.pekar, martin.chovanec}@tuke.sk

Liberios Vokorokos, Eva Chovancová
Peter Feciľak, Miroslav Michalko

Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice, Slovakia
e-mail: {liberios.vokorokos, eva.chovancova,
peter.fecilak, miroslav.michalko}@tuke.sk

Abstract. This paper explores the problem of processing the immense volume of measurement data arising during network traffic monitoring. Due to the ever-increasing demands of current networks, observing accurate information about every single flow is virtually infeasible. In many cases the existing methods for the reduction of flow records are still not sufficient enough. Since the accurate knowledge of flows termed as "heavy-hitters" suffices to fulfill most of the monitoring purposes, we decided to aggregate the flow records pertaining to non-heavy-hitters. However, due to the ever-changing nature of traffic, their identification is a challenge. To overcome this challenge, our proposed approach – the adaptive aggregation of flow records – automatically adjusts its operation to the actual traffic load and to the monitoring requirements. Preliminary experiments in existing network topologies showed that adaptive aggregation efficiently reduces the number of flow records, while a significant proportion of traffic details is preserved.

Keywords: Network traffic monitoring, IPFIX, exporter, flow record, data reduction, adaptive aggregation, heavy-hitter, resource utilization

Mathematics Subject Classification 2010: 68M10, 68M12, 90B18, 90B20

1 INTRODUCTION

Traffic monitoring has a significant role in the design, management and optimization of present computer networks. In order to achieve the full-featured operation of the network it is essential to measure and evaluate various characteristics of the traffic. Nowadays, the most commonly used data measurement methods are based on collecting information about the network and its traffic at the level of flows.

Network monitoring by flow-level based measurement platforms – either implementing the NetFlow v9 [1] or IPFIX [2] protocols – is based on the analysis of information obtained from traffic properties and characteristics. Flow-level information has a wide range of use from analyzing the traffic of the network through anomaly detection up to ensuring QoS. Collecting flow-level information can provide meaningful information about the dynamics of network traffic as well. Further, it is also used for a variety of network monitoring tasks such as user and application monitoring, traffic engineering, capacity planning, accounting, security applications (IDSes and IPSes) and performance analysis. Despite its popularity, flow-level measurement is still surrounded by several issues. As networks are continuously growing in

1. size,
2. connected users and
3. the volume of transmitted data (traffic),

their operation and management are becoming more and more complex. In consequence, current flow-based network monitoring systems generate a huge volume of measurement data what represents one of the most critical issues from the view of both data processing (analysis) and interpretation (visualization) [3].

In the following sections we provide a formal as well as informal description of our systematic approach that have finally led to the design of a yet another adaptive aggregation method. They discuss several methods/techniques and phenomena by the combination of which we achieved the aggregation of flow records and its adaptability to the traffic character.

1.1 State-of-the-Art

Over the past decades many approaches were taken to create a unified standard for monitoring the traffic which flows through the network. In consequence, various techniques and methods were proposed. All of these methods have their advantages and disadvantages. Some of them may excel in data reduction, but, on the other hand, the obtained information may be less detailed and have coarse granularity. On the contrary, other techniques may provide fine granularity, but data reduction may not be sufficient enough. Generally, most of the techniques have been incorporated into unified standards for retrieving network device- and traffic-specific information. Therefore, they are usually associated with a standard or protocol.

The most commonly used protocols in network traffic measurement are *counters* (SNMP counts [4]), *flow-level information* (Netflow/IPFIX [1, 5, 2]) and *sampled flow* (sFlow [6]).

Although SNMP counts are simple and lightweight to process, they do not provide enough details about the network traffic. When details about the traffic semantics are required, three main technologies come into consideration: *packet capture*, *sFlow* and *NetFlow/IPFIX*. Some of them affect the data reduction more and some less. Although NetFlow and IPFIX are surrounded by some challenges, their benefits far outweigh their shortcomings and in comparison with other approaches they provide more flexibility. Thus, collecting flow-level information is currently the most preferred way to perform measurements. It provides measurement data at a relatively high aggregation level while a considerable portion of traffic semantics needed for various network monitoring tasks is still retained. However, gathering flow-level information has to face some issues. The most emerging one is the volume of measurement data [7, 3]. In general, the main aim of the network is to ensure the smooth and fast transfer of traffic data. If the network has to deal with measurement data at the very same time, it can easily cause over-utilization of various network entities (e.g. inter-networking devices) and the network's operation. For example, if the individual inter-networking devices are involved in forwarding of measurement data between the observation point(s) and the monitoring system(s), their performance can be adversely affected. Similar situation occurs in case of routers which besides capturing packets are involved in the creation and export of flow records.

A recent approach to programmable networks is the Software Defined Networking (SDN) architecture, which is aimed, besides other things, at this issue. As described in [8], the main idea behind SDN is to allow developers to rely on network resources in the same easy manner as they do on computing or storage resources. This is reached by decoupling the control and data planes of the network, i.e., the network intelligence is logically centralized in software-based controllers (or control plane), and network devices become simple packet forwarding devices (or data plane) that can be programmed via an open interface (e.g. ForCES [9], OpenFlow [10, 11], etc.).

OpenFlow is currently the most commonly deployed Software Defined Networking (SDN) technology. Since the source code of the software running on the switches is usually inaccessible nor can be modified, it is difficult for the science-research community to test new ideas in current environments. OpenFlow was proposed to standardize the communication between the switches and the software-based controller in an SDN architecture, thus enabling researchers to test new ideas in a production hardware. It provides means to control a switch without requiring the vendors to expose the code of their devices. In the OpenFlow architecture the software-based controller is responsible for managing the forwarding information of one or more switches; while the hardware only handles the forwarding traffic according to the rules set by the controller [11].

Since OpenFlow separates the control plane and data plane of networking devices [11], it should be considered – as also suggested in [12] – a flow-based configura-

tion technology for packet forwarding devices, rather than a flow export technology. However, although it was not specifically developed for tasks related to data export and network monitoring, according to [13], flow-level information available within the OpenFlow control plane (e.g. packet and byte counters) was recently used for performing network measurements as well.

1.2 IPFIX-Based Measurement Platforms

At present, the vast majority of monitoring tools are performing the measurement of flow-level information using either the NetFlow [1] or the IPFIX [2] protocol. Since we expect IPFIX to be the industry standard for flow monitoring in the near future, and considering the fact that between IPFIX and NetFlow is just a slight difference, in the following we will analyze the process of network traffic monitoring in the context of the IPFIX specification.

IPFIX defines a format and a protocol for the export of information about IP flows. IP flow is defined as an unidirectional stream of IP packets identified by a common five-tuple (flow keys); specifically the protocol type, source IP address, destination IP address, source port and destination port. Basically, network traffic monitoring is based on the analysis of the exported information. The *properties* (e.g. the total number of bytes of all packets belonging to a certain flow) and *characteristics* (e.g. source IP address) of the flow are carried in *flow records*. The export of flow records represents a push-based mechanism, where the data are transmitted from the IPFIX exporter(s) to the IPFIX collector(s) over either the TCP, UDP or the SCTP protocol. Actually, the exporters and the collectors are the essential components of any IPFIX-based measurement platform.

Exporter is a device which basically hosts two processes: the *metering* and the *exporting*. Each of these processes can have one or more instances. In general, each exporter sends flow records to one or more *collectors*. The flow records are generated by the metering process(es).

The architecture of the exporter along with the metering and exporting processes is shown in Figure 1. The inputs for flow record generation are the packets themselves. It logically follows that the essential task of the metering process is packet capture. Its further optional functions include timestamping, packet selection (sampling and filtering) and classification.

Another important function of the metering process is maintaining the flow records. Tasks related to the maintenance of flow records include their creation, update, detection of flow expiration, passing the flow records to the exporting process and their removal. In practice, these tasks are performed using a flow cache.

The exporting process is situated a layer higher (see Figure 1). It provides an interface between the metering process(es) and the collecting process(es). In simple terms, it sends (exports) the flow records obtained from the metering

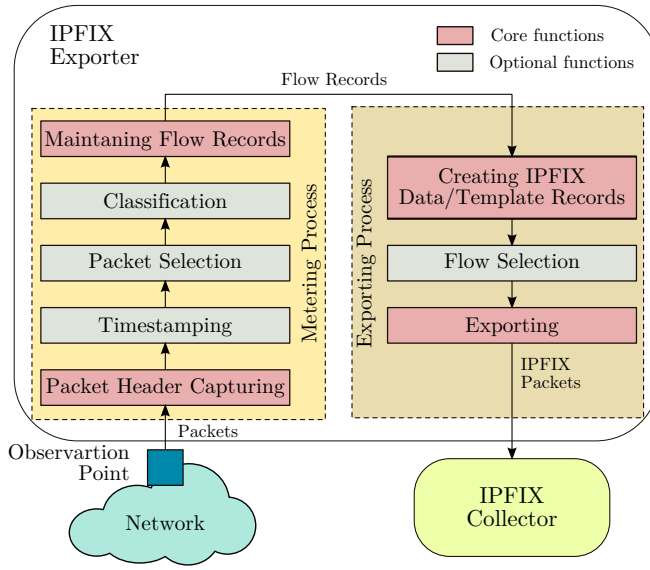

Figure 1. The general architecture of the exporter

process(es) to one or more collectors. The method of exporting the flow records is defined by the IPFIX protocol. Its further functional block is *flow selection* that using sampling or filtering can provide further reduction of data destined for export. This functional block of the exporting process is optional.

The metering process generates the flow records on the basis of the *flow keys*.

Flow keys are used to determine the conditions for creating flow records. In general, flow keys are information elements (IEs) [5] that define IP flows. The most commonly used information elements that serve as flow keys for flow generation are the sourceIPv4Address, destinationIPv4Address, sourceTransportPort, DestinationTransportPort and protocolIdentifier[1]. However, some information elements, for example those which belong to the timestamp and counter groups, cannot serve as flow keys.

Collector is a device which hosts a collecting process. The collecting process receives flow records from one or more exporting processes [5]. The main goal of the collector is to extract the measured properties and characteristics of the flow from the flow records. For efficiency, this information is stored and carried in information elements.

[1] Note that the notation of information elements in this work are in conformity with the IPFIX information model [5].

How the flow records are exported to the collector(s) is defined by the IPFIX protocol. Actually, they are transmitted by two kinds of information: *templates* and *data*. In simple terms, flow records are carried in data records and the structure of these data records is defined by the templates. It follows from the fact that traffic information depends on the purpose of the measurement and the network structure.

The information extracted from the data records can be stored in a *database* and/or directly sent to one or more external *evaluating* entities by mechanisms such as the Analyzer–Collector Protocol (ACP) introduced in one of our previous contributions [14]. Given the base functionality of the external entity (i.e. analysis), in the following we will refer to it as *analyzer*.

Analyzer provides further processing and analysis of the information about flows. More comprehensive analyzers also provide a GUI for both the visualization of the information obtained from the database/collector and the management of the architecture's lower components (i.e. exporter and collector). However, the analyzer itself is not a part of the IPFIX specification. For this reason, neither its requirements and functionalities, nor the communication principles between the analyzer and the other components of the IPFIX-based metering tool are limited.

In conclusion, monitoring and analyzing the network traffic based on the IPFIX protocol, as depicted in Figure 2, can be split into the following steps:

1. The information obtained from the captured packets after timestamping, sampling, classification, etc., are encapsulated into IPFIX messages and sent from the exporter(s) to the collector(s).
2. In the collector, after parsing the currently obtained template/data record, the obtained flow-level data are stored in the database of the metering platform and/or sent directly to the analyzer.
3. The analysis over the flow-level information is performed by an analyzing application. For example, the data obtained from a database can be processed and visualized in a form of plots. Obviously, these plots will vary according to the wanted type of analysis.

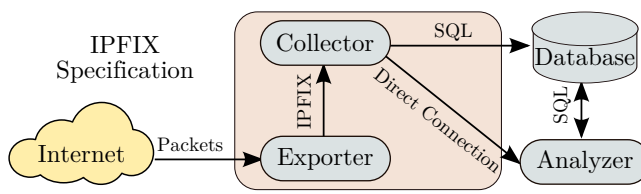

Figure 2. General architecture of a monitoring platform implementing IPFIX

1.3 Problem Statement

When too many flows are present in the traffic, IPFIX-based measurement platforms [15, 16] have to deal with several issues. The root cause of the issues is concerning the exporter(s), however, the immense volume of measurement data (flow records) also has a high impact on the evaluating processes. The issues which can rise during measurements are the following:

- Due to the connectionless nature of the UDP protocol, its use is avoided in many cases. Especially when the loss of information is not affordable (e.g. anomaly detection). It logically follows that TCP can provide a solution. However, when TCP is used and the export rate is too high, due to the (expected) overload of the collector, network congestion may occur between the exporter(s) and the collector(s). In consequence, the performance and accuracy of the evaluation processes might be significantly reduced.
- When using either TCP or SCTP, another issue represents their congestion avoidance mechanism. When the export rates are limited by congestion avoidance, the flow records can overwhelm the exporter. The expectable overflow of the flow cache (buffer) can subsequently result in

  1. the overutilization of the exporter's resources,
  2. the subsequent (radical) degradation of the flow records' accuracy and
  3. even in the exporter's crash.

  In network monitoring, the last issue represents a worst case scenario and its prevention is highly desirable in all circumstances.
- Even if there was a pair of exporter and collector capable of serving the problem-free export of flow records, the immense volume of data still represents an issue in the analyzer. With respect to the results of the experiments stated in [7], the demands related to the storage, processing, analysis, evaluation and visualization of the flow records proportionally grow with their number.

Flow records are obviously not only an issue of two parties (i.e. exporter and collector). Their analysis, evaluation and interpretation in the analyzer represent also a demanding task. We can conclude that the core of the problems is the huge number of flow records, which the exporter has to process and how it provides the data for the upper layers. Since it is unable to measure the traffic on a per-flow basis, in order to optimize the monitoring systems, the measurement data amount has to be reduced.

1.4 Related Work

Initial efforts [17, 18] to monitor the network on a per-flow basis appeared to be unscalable. The cost and speed of the memory required for a simple measurement

of a large number of flows and subsequent generation of flow records proved to be prohibitive. In response, packet and flow sampling techniques started to be used to reduce the complexity of the metering and monitoring devices [19, 20, 21]. Since these methods use sampling, they are exposed to a trade-off between monitoring accuracy and limited resources (e.g. memory size, CPU speed).

The paper by Dressler and Münz [22] introduces an aggregation mechanism for efficient and flexible use in flow accounting scenarios. Their approach is controlled by aggregation rules that allow adapting to several application requirements. However, their proposed approach counts only with the requirements of the monitoring purpose. They do not adapt the aggregation to the traffic behavior.

Hu et al. [23] proposed an entropy-based adaptive flow aggregation algorithm. They claim that their mechanism provides a relief from DoS attacks and other security breaches. Unlike the mechanism designed by Dressler and Münz, this algorithm does not take into account that the arbitrarily defined flow aggregates usually do not meet the requirements of the monitoring purpose.

The most accurate version of the document [24] extending the IPFIX standard provides a common implementation-independent basis for an intermediate component between the exporter and collector to handle flow aggregation. However, since the export of flow records between the exporter and the mediator is performed over the same protocols as between the exporter and the collector (i.e. TCP, UDP, SCTP), this standard still does not address the issues described in Section 1.3. As we already stated, the issue of immense volume of flow records has to be solved at the lowest layer of the IPFIX-based measurement platform. Passing "the core of the problem" to the higher layers is not a good strategy to solve the issue.

A promising way how to deal with large data sets is to deploy various data reduction methods in the storage system of the monitoring tool. Although this solution – as described in one of our previous works [7] – can bring positive results, in a long term – as described in our further work [3] – they still do not represent an appropriate workaround. Mainly due to the fact that the network has a lot more devices than the monitoring system. This results in an incomparable difference between their computation resources, i.e. the resources for traffic generation and traffic measurement. Therefore, the issue of immense volume of flow records have to be solved at the lowest layer of the IPFIX-based measurement platform, i.e. in the exporter. Passing "the core of the problem" to the higher layers is not a good strategy.

We can conclude that all the aforementioned approaches provide a solution to the immense volume of measurement data at an acceptable level. A common denominator of these approaches is that even if they adapt their operation to several conditions, they do so only to one at a time. Intuitively, we can expect that increasing the number of conditions taken into consideration during the adaptation process should bring further improvements.

2 ADAPTIVE AGGREGATION OF FLOW RECORDS

Monitoring mechanisms, whether it comes to the utilization of resources or to the "volume of the measurement data/required granularity of information" ratio, often perform inefficiently. It is mainly due to the absence of capabilities by which they could adapt their "behavior" to the actual state of the network. With no doubt, given the ever changing character of present network traffic, this adaptation is difficult to achieve. Even if adaptability is achieved, like in case of the approaches from Section 1.4, it is usually performed according to only a single criterion/condition. However, if there was a way to adapt the operation of the monitoring tool not only to the traffic character but also to the requirements of the monitoring purpose, the immense volume should not be longer an issue at higher layers (i.e. collector, analyzer). This is the basic idea that evolved to our proposed method – the adaptive aggregation of flow records.

2.1 Aggregation of Flow Records

The general architecture of the exporter (see Figure 1) provides several options where the number of flow records can be reduced, specifically:

- two in the metering process: sampling and filtering (denoted in Figure 1 as Packet Selection);
- and two in the exporting process: sampling and filtering (denoted in Figure 1 as Flow Selection).

As we can see, sampling and filtering techniques can be applied at two different layers on data of two different character. While sampling and filtering represent in the metering process a packet selection task, in the exporting process they are a task of flow selection. In addition, there are several other methods for data reduction whose implementation in the exporter would not violate the IPFIX specification (their most appropriate location is between the metering and exporting processes). However, methods such as K-means or dimensionality reduction (e.g. Cluster Analysis, Principal Component Analysis, etc.) are – due to their complexity and computation demands – not suitable for implementation in the exporter [3]. This leaves us, as suggested in the IPFIX specification [2], with three further data reduction techniques: sampling, filtering and aggregation. Although each of them have their advantages and disadvantages, aggregation provides the most comprehensive solution for data reduction. It combines the advantages of both, sampling and filtering. Indeed, while it can efficiently reduce the number of flow records (like sampling); using subnet masks, it also provides a way to focus only on a specific measurement target (like filtering). Its main advantage is that unlike sampling, which discards an uncertain number of packets, the aggregated flow record reflects all the flow properties. However, we can often observe that even with aggregation, the number of measured flow records within a relatively small time interval is still

too high [20, 23, 25]. Therefore, in order to achieve the reduction of flow records, we had to make a shift away from the classical way of their aggregation.

2.2 Aggregation of Non-Heavy-Hitter Flows

Examination of various phenomena in flows that can be statistically described by either self-similarity, first-order similarity (long-range dependence) or heavy-tailed distribution have been an objective of several research activities [26, 27]. A common observation which can be deduced from these research activities is that a very small percentage of flows carry the main part of the traffic (in bytes). We generally refer to these flows as *heavy-hitter flows*.

Heavy-hitters and non-heavy-hitters enable us to examine the dynamics and semantics of network traffic from a new perspective. Their main advantage is that the differentiation of flows into two main classes (i.e. heavy-hitters and non-heavy-hitters) can radically contribute to the reduction of the volume of flow records. In addition, several research activities [20, 28, 21, 27] report that for many monitoring purposes the accurate knowledge of heavy-hitters is still sufficient enough. These monitoring purposes include anomaly and attack detection, scalable differentiated services, usage-based pricing and accounting, making decisions about network upgrades and peering. Therefore, instead of aggregating flow records in a classical manner, we rather performed this critical tasks of the exporter with respect to the heavy-tailed nature of network traffic.

From the well-know and commonly used flow types [27] we can deduce that the elephant flow[2] is the one for which holds that the smallest percentage of its flows accounts for the largest percentage of the transmitted data. It logically follows that in case of mouse flows, the largest percentage of the flows accounts for the smallest percentage of the transmitted data. Therefore, from the view of data reduction, the highest level of compression of flow records can be achieved by the aggregation of mouse flows. Considering the all above, we proposed the aggregation of flow records on the basis of the separation of flows into elephant flows and mouse flows:

We keep the heavy-hitters (elephants) in their original form and rather aggregate the non-heavy-hitters (mice); where

elephant flow is a flow in which the total number of transferred data expressed in bytes $\left(n_{td}^{f}\right)$ is larger than or equal to a predefined *threshold* T, i.e.

$$\text{elephant} \stackrel{def}{=} n_{td}^{f} \geq T; \text{ and} \tag{1}$$

[2] The method of assigning the names to the flow types depends on the characteristics these flows exhibit in network traffic [28].

mouse flow is a flow in which the total number of transferred data expressed in bytes $\left(n_{td}^{f}\right)$ is less than a predefined *threshold* T, i.e.

$$\text{mouse} \stackrel{def}{=} n_{td}^{f} < T. \tag{2}$$

It is obvious that the evaluation of Equations (1) and (2) requires an accurate knowledge of *the total number of transferred data* $\left(n_{td}^{f}\right)$ of each flow. Fortunately, the *octetTotalCount* information element of the IPFIX information model [5] provides exactly such a measure, according to which:

octetTotalCount is the total number of octets (bytes) in incoming packets for a given flow at an observation point.

In conclusion, flows meeting the following condition will be aggregated:

$$\textit{aggregate the flow record} \stackrel{def}{=} octetTotalCount < T. \tag{3}$$

From the perspective of resource utilization, the accurate estimation of *threshold* T is a critical task. Due to the dynamic nature of the traffic its value cannot be estimated by performing a one-time experiment, because – although it would be suitable for the identification of mice at one point in time – it will be unsuitable at another point in time. This led us to its adaptation to the traffic load.

2.3 Adapting the Aggregation to the Traffic Character

The proposed aggregation adaptability is based on the resource utilization of the exporter. For this purpose we track two parameters, the exporter's CPU utilization and memory utilization. The number of actually processed packets are directly proportional to these two parameters. It means that if the traffic increases, the number of captured packets grows as well. As a result, the exporter experiences an increase in the CPU and memory load. On the contrary, if the traffic decreases, since the number of captured packets in the exporter shrinks, the CPU and memory load decreases as well. We can therefore conclude that:

$$\textit{the number of processed packets} \propto CPU\ load\ (\%) \propto Memory\ load\ (\%). \tag{4}$$

As a result, we can define four different loads (characters) of the network traffic:

- *Weak traffic* – traffic utilizing 1/3 of the exporter's CPU and memory resources.
- *Moderate traffic* – traffic utilizing 2/3 of the exporter's CPU and memory resources.
- *Strong traffic* – traffic utilizing $3/3 - k$ part of the exporter's CPU and memory resources, where the subtraction of k determines the maximum resource utilization while the exporter still reliably processes all the captured packets.

- *Critical traffic* – traffic utilizing 3/3 of the exporter's CPU and memory resources. In other words, this traffic is utilizing the exporter to the maximum of its resources, resulting in packet loss.

These individual network traffic loads are depicted in Figure 3.

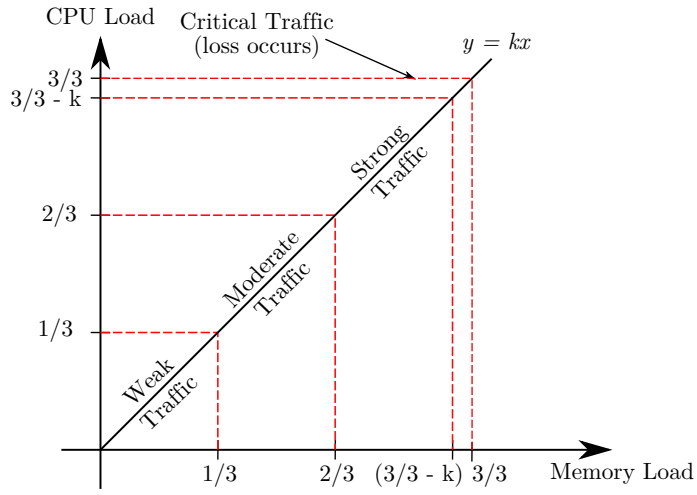

Figure 3. Traffic loads defined by the resource utilization of the exporter

Using these traffic types, instead of estimating the real character of the traffic we rather estimate how it is perceived by the exporter. Since the direct proportionality (Equation (4)) works in both directions, we can observe the network traffic character by estimating the resource utilization of the exporter. If we consider the resource utilization increases (decreases) to be a continuous process, we can define several condition (state) changes by its discretization into time slices using regularly spaced intervals. This allows us to adapt the threshold – for the separation of the flows into heavy-hitters and non-heavy-hitters – to the actual state of traffic. Considering all the above, the process of adapting the threshold to the individual states of the traffic consists of the following steps:

1. Observe the CPU and memory[3] utilization of the exporter in discrete time intervals. For each time slice compute the value by averaging the CPU and memory load. As a result, we get a *set of n observations* $\mathcal{O}$ of the resource utilization of the exporter at each *time slice* t; i.e.

$$\mathcal{O} = \{o_t, o_{t+1}, \ldots, o_{t+n}\} \qquad t = 0, 1, \ldots, k \tag{5}$$

[3] Note that in case of the exporter, by over memory utilization we mean the load of the cache in which it holds the captured packets, i.e. packet cache.

where each *observation* o is computed by averaging the CPU and memory load at time t, i.e.

$$o_t = (cpu_t + memory_t)/2. \tag{6}$$

2. At the initialization of the aggregation process, compute the *average resource utilization* (ARU) from all the observed values since the last process of aggregation; i.e.

$$ARU = \frac{\sum_{i=0}^{k} o_{t+i}}{k}. \tag{7}$$

3. Given the value of ARU, determine the pertaining *traffic character* (TCh), i.e.

$$TCh = \begin{cases} \text{Weak} & \text{if } ARU \in \{1, 2, \ldots, 32\}, \\ \text{Moderate} & \text{if } ARU \in \{33, 34, \ldots, 65\}, \\ \text{Strong} & \text{if } ARU \in \{65, 68, \ldots, 98\}, \\ \text{Critical} & \text{if } ARU \in \{99, 100\}. \end{cases} \tag{8}$$

In words, determine the pertaining actual *traffic character* (TCh) depending on the actual *average resource utilization* of the exporter (ARU).

4. Adjust the value of the *threshold* (T) for the separation of elephant and mouse flows to the actual *traffic character* (TCh) according to the following criteria:

if TCh_t or TCh_{t+1} == Critical then

$$T = \begin{cases} 2T_d & \text{if } TCh_t < TCh_{t+1}, \\ \frac{T_d}{2} & \text{if } TCh_t > TCh_{t+1} \end{cases} \tag{9}$$

else

$$T = \begin{cases} T + (|ARU_t - ARU_{t+1}|)\,\%\text{ of } T_d & \text{if } TCh_t < TCh_{t+1}, \\ T - (|ARU_t - ARU_{t+1}|)\,\%\text{ of } T_d & \text{if } TCh_t > TCh_{t+1} \end{cases}$$

where T_d is the *default threshold* determined by pilot measurements. In words, if there was a transition where either the *traffic character at the previous process of aggregation* (Tch_t) or the *current traffic character* (TCh_{t+1}) was *Critical*, set the *threshold* T to the double or halve of the *default threshold* (T_d) depending on the direction of this transition. Otherwise, i.e. if Tch_t or Tch_{t+1} was *Weak*, *Moderate* or *Strong*, if the *traffic character at the previous process of aggregation* (Tch_t) was smaller than the *current traffic character* (TCh_{t+1}) (i.e., the number of packets increased), T will be increased by as many percent of the *default threshold* (T_d), by as many the *average resource utilizations* (ARU) in time t and $t+1$ differ. However, if TCh_t was larger than TCh_{t+1} (i.e. the number of packets decreased), T will be decreased by as many percent of T_d, by as many the ARUs in time t and $t+1$ differ.

5. Store the current values of the actual *traffic character* (TCh) and the actual *average resource utilization* (ARU) for the next iteration of this process.

All we need to do is to set an initial *threshold* and an estimate of the resource utilization of the exporter at the beginning of the measurement. The aforementioned procedure will automatically adjust the *threshold* to the traffic character in which the individual packets of the flows are captured. In addition, since the individual traffic characters are computed via averaging, our method can also deal with various peaks in the traffic (i.e. they will be averaged out).

2.4 Adapting the Aggregation to the Purpose of the Monitoring

Aggregation obviously causes some information losses. As a result, the information provided by aggregated flow records has a coarse granularity. On the other hand, our aim is to retain as many details of the network traffic as possible. This makes a trade-off between the granularity of information and the volume of flow records. Although our approach neglects the informational value of mouse flows, during the aggregation we still want to preserve as many details about non-heavy-hitters as possible. With respect to the nature of aggregation, this can be achieved only if we take the purpose of monitoring into account. If we know which information elements provide the most valuable information for the monitoring, all we need to do is to take them into account during the process of aggregation. If we order the information elements serving as flow keys [2] from the lowest to the highest according to a ranking indicating the significance of the information element in the context of the network monitoring purpose, we get n different levels of aggregation:

1. The first level of aggregation is performed over the information element with the lowest ranking. As a result, the information carried by the aggregated flow records has the "least coarse" level of granularity[4].
2. The second level of aggregation is performed over the information element with the lowest ranking among the remaining information elements (i.e. the next information element). As a result, the information carried by the aggregated flow records has a coarser level of granularity.
3. The same procedure is repeated iteratively until it gets to the last information element.
4. The last level of aggregation is performed over the n^{th} information element having the highest ranking. As a result, the information carried by the aggregated flow records has the coarsest level of granularity.

This procedure[5] is illustrated in Figure 4.

In consequence, even if we focus only on heavy-hitter flows, this method allows to aggregate efficiently among the mouse flows. As a result, we can further separate mouse flows into different classes that preserve the details of the traffic at various levels of granularity.

[4] The finest level of granularity provides the not aggregated heavy-hitter flow records.

[5] Note that this approach represents a generalized form of the *Gradual Flow Key Reduction* method [29].

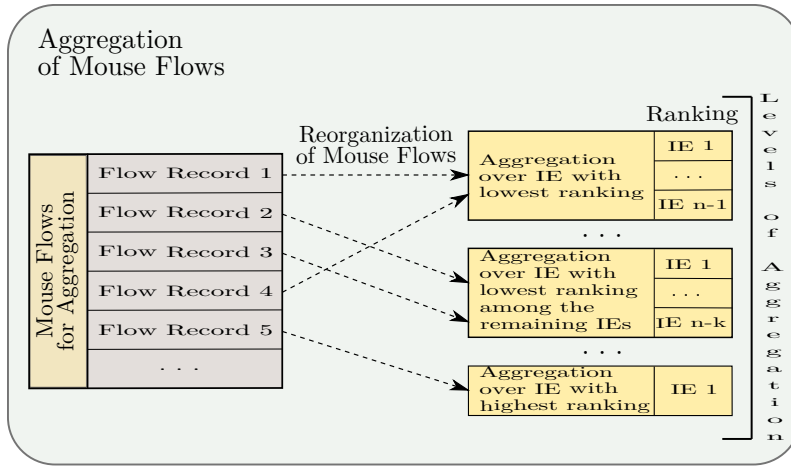

Figure 4. Adapting the aggregation to the purpose of the monitoring based on [29]

2.5 Triggering the Process of Aggregation

The organization of packets into flows is a continuous process. Depending on whether there is a record in the flow cache pertaining to the actually processed packet or not, the existing flow record is updated or a new one is created. Obviously, the value of *octetTotalCount* of those packets that belong to the same flow is conditioned by these creations and updates. However, our approach for the separation of heavy-hitter and non-heavy-hitter flows is based exactly on this value (see Equation (3)). In consequence, the aggregation frequency must not be too low.

This is due to the fact that during a too short period of time the metering process cannot capture enough information about the flows to separate them correctly into elephants and mice. As a result, all the flows will be identified as mouse flows and thus, they will be all aggregated; even those that are in real elephant flows. In addition, a constantly running aggregation process can cause overutilization of the exporter. Therefore, instead of carrying out aggregation at every point in time, we decided for its execution in particular time periods. For this purpose we define an *aggregation frequency* variable.

Aggregation frequency (A_f) is the measure of time period expressed in milliseconds (ms) or the several fractions of it (e.g. microseconds, nanoseconds, etc.) after which the aggregation of non-heavy-hitter flows takes place.

In other words, A_f represents the rate at which the aggregation is performed. The determination of its value, as in case of the *threshold*, highly depends on the character of the traffic. It means that "weak" network traffic requires a different value of A_f than "strong" network traffic. For example, while in the case of weak traffic, aggregation performs properly at higher *aggregation frequency* values (e.g.

$A_f = 500\,\text{ms}$), in the case of strong network traffic it provides accurate results at lower *aggregation frequency* values (e.g. $A_f = 100\,\text{ms}$). However, since the *threshold* is already automatically adapted to the traffic character by the method described in Section 2.3, there is no need to adjust also the aggregation frequency (A_f). In simple words, instead of adapting the value of A_f we rather adapt the *threshold*, by which we should achieve the same result.

3 PRELIMINARY EXPERIMENTS

The module of adaptive aggregation was implemented in the `BEEM` component of the `SLAmeter` IPFIX-based measuring platform [16] as a full-featured functional block between the metering and the exporting processes. The aim of the experiments was to verify the functionality of the SLAmeter with and without the proposed method. For this reason we generated the same artificial traffic by several packet generators.

The accurate operation of adaptive aggregation required some conditions to be satisfied. The individual information elements [5] that were set as flow keys for the generation of flow records were the following: protocolIdentfier (highest rank), sourceTransportPort, destinationTransportPort, destinationIPv4Address, sourceIPv4Address (lowest rank). The pilot measurement showed that the `BEEM` processed the packets in every 35 000 ns. With respect to the predefined size of the allocated memory for the flow records (flow cache = 8 MB), the value of A_f was set to 250 ms in the `BEEM`; where A_f is the time period after which the aggregation of non-heavy-hitter flows takes place. During this time interval, the `BEEM` was able to hold approximately 7 150 records in its flow cache at a time. Measurement results also showed that the total value of transferred octets of almost 92 % of the identified flows was less than 10 000. In other words, during this pilot measurement the `BEEM` did not capture any flow having more than 10 000 B (*bytes*) of transferred data till its passive expiration. Considering all the above, the initial threshold T for the identification of elephant and mouse flows was set to 10 000 octets in the `BEEM`, i.e. $octetTotalCount < 10\,000$.

Assuming these prerequisites were satisfied, the process of adaptive aggregation consisted of the iterative execution of the following two phases:

1. the adaptation phase, in which the threshold T was adjusted according to the method described in Section 2.3; and
2. the aggregation phase, in which the individual flow records were aggregated according to the method described in Section 2.4.

The results of the experiments are summarized in Tables 1 and 2.

Without adaptive aggregation, as shown in Table 1, approximately 120 000 flows (sessions) were generated and as a result created approximately 300 000 flows records in the flow cache. However, since the flow cache after a specific period of time (around time 10:30:00) was utilized to its maximum, an undetermined number of packets was not organized into flow records. Therefore, the values in Table 1 cannot be considered as complete and have a certain bias. Moreover, the average load of

the flow cache was over 90 % during the whole measurement. The flow-rate plot pertaining to this measurement is shown in Figure 5 (denoted with red). From the plot we can see that the average number of flows per a second was between 30 and 50.

Measurement Characteristics	Results
Number of generated packets	1 950 448
Number of flows	121 641
Number of transferred data (in MB)	1 455.1
Total number of flow records	281 537
Average load of Flow Cache	90 %

Table 1. Results without adaptive aggregation

With adaptive aggregation the threshold T was automatically adjusted according to the method described in Section 2.3. The summary of the measurement is shown in Table 2. As we expected, the average load of the flow cache was lower, (i.e. around 20 %) during the whole measurement. The flow-rate plot pertaining to this measurement is shown in Figure 5 (denoted with blue color). From the plot we can see that the average number of flows per a second was between 7 and 15. The plot perfectly emphasizes the coarse granularity of the observed information resulting from the aggregation. In addition, adaptive aggregation also handled the burst in the flows that caused the erroneous operation of `BEEM` during the measurement without adaptive aggregation.

Measurement Characteristics	Results
Number of generated packets	1 950 448
Number of flows	3 460
Number of transferred data (in MB)	1 455.1
Total number of flow records	6 670
Average load of Flow Cache	20 %

Table 2. Results with adaptive aggregation

Discussion

Adaptive aggregation radically reduced the number of flows. However, when we compared the error rate between the measurements, we found that there was a trade-off between the accurate identification of elephant/mouse flows and the load of flow cache. In numbers, although the load of the flow cache was around 20 % during the measurement with adaptive aggregation, the error rate of the identification of mouse flows was around 12–14 %. The determination of the error rate was achieved by the reconstruction of the flows exported and stored in the database from the first

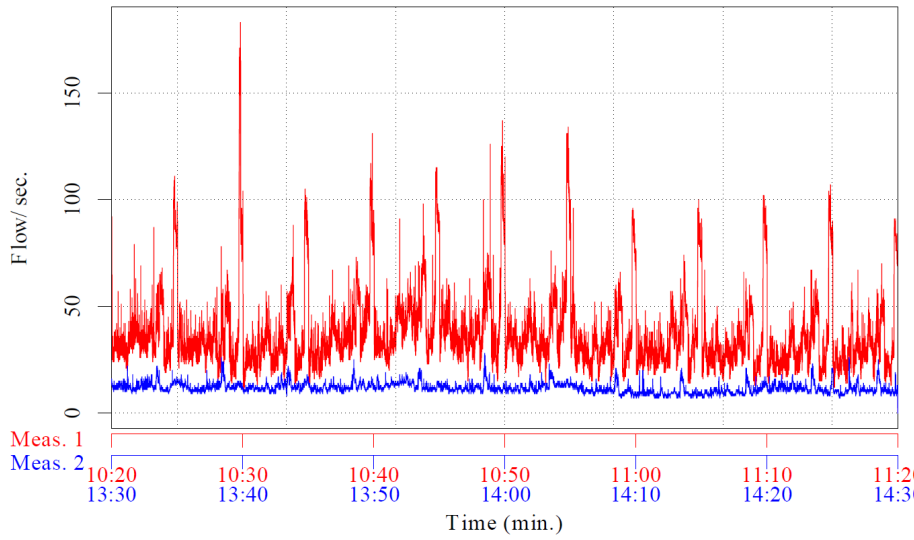

Figure 5. Combined flow-rate plot resulted from the measurements. Red colored flow-rate plot pertains to the measurement without adaptive aggregation (Meas. 1). Blue colored flow-rate plot pertains to the measurement with adaptive aggregation (Meas. 2).

measurement and compared with the results from the second measurement (time intervals with packet loss were discarded).

We can therefore conclude that aggregation involves a certain error rate into the measurements. However, when comparing this bias while all data are preserved and a certain data loss because of memory issues, this error rate can be considered acceptable. In summary, our approach has a significant impact on the monitoring systems that implement IPFIX for collecting information about IP flows. Since in case of the adaptation to the traffic character we did not consider the utilization of the link bandwidth, the accuracy of the measurements still highly depends on the initial value of the estimated threshold.

4 CONCLUSION AND FUTURE WORK

In this paper we introduced yet another approach whose aim is to address the issues related to the volume of measurement data produced during network monitoring. Its main contribution is the method of adaptive aggregation that adjusts its operation to the network character and to the purpose of monitoring. To the best of our knowledge, only a few methods adapts their behavior to more than one factor. The results presented in Section 3 proved that by adaptive aggregation we can efficiently reduce the number of measurement data and it has a positive impact on the measurement platform. We can therefore conclude that our proposed approach

considerably contributes to the research area of the measurement platforms based on the IPFIX protocol.

Since adaptive aggregation, due to the error rate, brings a certain bias into the measurement data, in the future, the method of adjusting the threshold for the identification of (non) heavy-hitters will be extended with the utilization of the link bandwidth as well. The proposed approach will be also examined in network monitoring scenarios with long duration. We will also review the possibility to implement dynamic Bayesian networks, by which the traffic could be classified according to the assumption that the probability distribution of different traffic types are far from each other in the same network.

Acknowledgments

This publication is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF. We support research activities in Slovakia/This project is co-financed by the European Union.

REFERENCES

[1] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC3954, 2004, doi: 10.17487/rfc3954.

[2] Claise, B.—Trammell, B.—Aitken, P.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC7011, 2013, doi: 10.17487/rfc7011.

[3] Pekár, A.—Chovancová, E.—Fanfara, P.—Trelová, J.: Issues in the Passive Approach of Network Traffic Monitoring. Proceedings of the 17th IEEE International Conference on Intelligent Engineering Systems (INES), 2013, pp. 327–332, doi: 10.1109/INES.2013.6632836.

[4] Case, J.—Fedor, M.—Schoffstall, M.—Davin, J.: Simple Network Management Protocol (SNMP). RFC1157, 1990, doi: 10.17487/rfc1157.

[5] Claise, B.—Trammell, B.: Information Model for IP Flow Information Export (IPFIX). RFC7012, 2013, doi: 10.17487/rfc7012.

[6] Phaal, P.—Lavine, M.: sFlow Version 5. Specification, sFlow.org, 2004.

[7] Vokorokos, L.—Pekár, A.—Ádám, N.: Data Preprocessing for Efficient Evaluation of Network Traffic Parameters. Proceedings of the 16th IEEE International Conference on Intelligent Engineering Systems (INES), 2012, pp. 363–367, doi: 10.1109/INES.2012.6249860.

[8] Nunes, B. A. A.—Mendonca, M.—Nguyen, X.-N.—Obraczka, K.—Turletti, T.: A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys and Tutorials, Vol. 16, 2014, No. 3, pp. 1617–1634, doi: 10.1109/SURV.2014.012214.00180.

[9] Haleplidis, E.—Salim, J. H.—Halpern, J. M.—Hares, S.—Pentikousis, K.—Ogawa, K.—Wang, W.—Denazis, S.—Koufopavlou, O.: Network Programmability with ForCES. IEEE Communications Surveys and Tutorials, Vol. 17, 2015, No. 3, pp. 1423–1440, doi: 10.1109/COMST.2015.2439033.

[10] Kang, M.—Kang, E.-Y.—Hwang, D.-Y.—Kim, B.-J.—Nam, K.-H.—Shin, M.-K.—Choi, J.-Y.: Formal Modeling and Verification of SDN-OpenFlow. Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation (ICST), 2013, pp. 481–482.

[11] Lara, A.—Kolasani, A.—Ramamurthy, B.: Network Innovation Using OpenFlow: A Survey. IEEE Communications Surveys and Tutorials, Vol. 16, 2014, No. 1, pp. 493–512.

[12] Hofstede, R.—Čeleda, P.—Trammell, B.—Drago, I.—Sadre, R.—Sperotto, A.—Pras, A.: Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. IEEE Communications Surveys and Tutorials, Vol. 16, 2014, No. 4, pp. 2037–2064.

[13] Yu, C.—Lumezanu, C.—Zhang, Y.—Singh, V.—Jiang, G.—Madhyastha, H. V.: FlowSense: Monitoring Network Utilization with Zero Measurement Cost. Passive and Active Measurement (PAM 2013). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7799, 2013, pp. 31–41, doi: 10.1007/978-3-642-36516-4_4.

[14] Pekár, A.—Révés, M.—Giertl, J.—Feciľak, P.: Overview and Insight into the MONICA Research Group. Central European Journal of Computer Science, Vol. 2, 2012, No. 3, pp. 331–343.

[15] Jakab, F.—Košč̆o, Ľ.—Potocký, M.—Giertl, J.: Contribution to QoS Parameters Measurement: The BasicMeter Project. Proceedings of the International Conference on Emerging eLearning Technologies and Applications, 2005, pp. 371–377.

[16] Pekár, A.—Feciľak, P.—Michalko, M.—Giertl, J.— Révés, M.: SLAmeter – The Evaluator of Network Traffic Parameters, Proceedings of the 10th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA), 2012, pp. 291–295, doi: 10.1109/ICETA.2012.6418318.

[17] Ramabhadran, S.—Varghese, G.: Efficient Implementation of a Statistics Counter Architecture. ACM SIGMETRICS Performance Evaluation Review (PER), Vol. 31, 2003, No. 1, pp. 261–271, doi: 10.1145/781027.781060.

[18] Shah, D.—Iyer, S.—Prabhakar, B.—McKeown, N.: Analysis of a Statistics Counter Architecture. Hot Interconnects 9, 2001, pp. 107–111, doi: 10.1109/HIS.2001.946701.

[19] Estan, C.—Keys, K.—Moore, D.—Varghese, G.: Building a Better NetFlow. ACM SIGCOMM Computer Communication Review (CCR), Vol. 34, 2004, No. 4, pp. 245–256.

[20] Estan, C.—Varghese, G.: New Directions in Traffic Measurement and Accounting. Proceedings of the 2002 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2002, pp. 323–336, doi: 10.1145/633025.633056.

[21] Lu, Y.—Wang, M.—Prabhakar, B.— Bonomi, F.: ElephantTrap: A Low Cost Device for Identifying Large Flows. Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects, 2007, pp. 99–108, doi: 10.1109/HOTI.2007.13.

[22] Dressler, F.—Münz, G.: Flexible Flow Aggregation for Adaptive Network Monitoring. Proceedings of the 31st IEEE Conference on Local Computer Networks, 2006, pp. 702–709, doi: 10.1109/LCN.2006.322180.

[23] Hu, Y.—Chiu, D.-M.—Lui, J. C. S.: Entropy Based Adaptive Flow Aggregation. IEEE/ACM Transactions on Networking, Vol. 17, 2009, No. 3, pp. 698–711.

[24] Trammell, B.—Wagner, A.—Claise, B.: Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol. RFC7015, 2013, doi: 10.17487/rfc7015.

[25] Cheng, G.—Gong, J.: Adaptive Aggregation Flow Measurement on High Speed Links. Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS 2008), 2008, pp. 559–563, doi: 10.1109/ICCS.2008.4737246.

[26] Papagiannaki, K.—Taft, N.—Bhattacharyya, S.—Thiran, P.—Salamatian, K.—Diot, C.: A Pragmatic Definition of Elephants in Internet Backbone Traffic. Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW '02), 2002, pp. 175–176, doi: 10.1145/637201.637227.

[27] Smith, R. D.: The Dynamics of Internet Traffic: Self-Similarity, Self-Organization, and Complex Phenomena. Advances in Complex Systems, Vol. 14, 2011, No. 6, pp. 905–949.

[28] Lan, K.-C.—Heidemann, J.: A Measurement Study of Correlations of Internet Flow Characteristics. Computer Networks, Vol. 50, 2006, No. 1, pp. 46–62.

[29] Irino, H.—Katayama, M.—Chaki, S.: Study of Adaptive Aggregation on IPFIX. Proceedings of the 7th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), 2008, pp. 86–91.

Adrián Pekár graduated at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice, Slovakia in 2011. Since then, his scientific research has been focused on the optimization of measurement platforms based on the IPFIX protocol. He defended his Ph.D. thesis in the field of network traffic characteristics measurements and monitoring in 2014. Recently, his scientific research was extended to investigate also the issues related to monitoring and virtualization of cloud networks. His area of interest includes QoS, IPFIX, network traffic management and engineering, cloud computing and virtualization.

Martin Chovanec received his Engineering degree in Informatics in 2005 from the Faculty of Electrical Engineering and Informatics, Technical University of Košice. In 2008 he received his Ph.D. degree at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice and his scientific research was focused on network security and encryption algorithms. Currently, he is Director of the Institute of Computer Technology of the Technical University of Košice.

Liberios Vokorokos graduated (M.Sc.) with honors at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice in 1991. He defended his Ph.D. thesis in the field of programming devices and systems in 2000. In 2005 he became Professor of Computer Science and Informatics. Since 1995 he has worked as an educationist at the Department of Computers and Informatics. His scientific research focuses on parallel computers of the Data Flow type. He also investigates the issues related to the complex systems diagnostics. He is the Dean of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice.

Eva Chovancová graduated (M.Sc.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2009. She defended her Ph.D. thesis in the field of computers and computer systems in 2012; her thesis title was "Specialized Processor for Computing Acceleration in the Field of Computer Vision". Since 2012 she has worked as Assistant Professor at the Department of Computers and Informatics. Her scientific research is focused on the multicore computer architectures.

Peter Feciľak graduated (M.Sc.) at Department of Computers and Informatics at Faculty of Electrical Engineering and Informatics, Technical University of Košice in 2006. In 2009, he finished his Ph.D. studies at the named department with the focus on optimization of computer networks. Currently, he is working as an employee of DCI, FEI, Technical University of Košice. His current teaching and research interests are computer networks, network monitoring, quality of services and smart energy systems.

Miroslav Michalko received his Ph.D. in informatics from the Technical University of Košice (TUKE), Slovakia. For more than 10 years he is a member of a well recognized research institution – the Computer Networks Laboratory at Department of Computers and Informatics (DCI) of TUKE. Currently he is Assistant Professor at DCI TUKE and gives lectures in the field of computer networks. His research includes multimedia content delivery, video streaming services, web and cloud services, innovative teaching and learning techniques and IoE/IoT solutions.