Cent. Eur. J. Comp. Sci. • 2(3) • 2012 • 331-343
DOI: 10.2478/s13537-012-0013-9

VERSITA

Central European Journal of **Computer Science**

Overview and insight into the MONICA research group

Review Article

Adrián Pekár*, Martin Révés†, Juraj Giertl‡, Peter Feciľak§

*Computer Networks Laboratory,
Technical University of Košice,
Letná 9, 04200 Košice, Slovakia*

Received 09 February 2012; accepted 17 August 2012

Abstract: This paper deals with the activities of the MONICA research group of the Computer Networks Laboratory at the Technical University of Košice in Slovakia. MONICA stands for the Monitoring and Optimization of Network Infrastructures, Communications and Applications. Our main areas of interest are related to development of the BasicMeter tool built in conformity with the IPFIX architecture. In this paper we give an overview of our solution and proposal of several enhancements over the scope of IPFIX, such as time synchronization for the One-Way Delay measurements, the protocol for direct communication with the analyzing application, the centralized management of the monitoring platform, and support for real-time monitoring.

Keywords: network monitoring • network traffic • monitoring tool • IPFIX • One-Way Delay

1. Introduction

Network traffic monitoring plays a key role in the operation of computer networks. There are plenty of motivating factors for network monitoring with various purposes in different application domains such as usage-based accounting, SLA evaluation, security analysis, traffic engineering, traffic parameters analysis and their optimization as a supporting mechanism for the implementation of QoS needed for certain services, etc. The importance of network monitoring is obvious.

We in the MONICA research group in the Computer Networks Laboratory at the Technical University of Košice in Slovakia have worked for nearly ten years in the area of network monitoring. All it began with a short stay of our colleagues in the Fraunhofer Institute for Open Communication Systems in Berlin. We then developed our first monitoring tool based on the NetFlow specification [8] which we later extended with the support of the IPFIX protocol [2]. A few years later we founded the MONICA Research Group with a focus on the monitoring and optimization of network infrastructures, communications and applications. All MONICA activities are in some way connected to network monitoring. We follow several goals:

* *E-mail: adrian.pekar@tuke.sk (Corresponding author)*
† *E-mail: martin.reves@tuke.sk*
‡ *E-mail: juraj.giertl@tuke.sk*
§ *E-mail: peter.fecilak@tuke.sk*

- Development of tools and methods for obtaining IP flow information – the main goal is the development of the BasicMeter tool in conformity with the IPFIX [19] and PSAMP [22]. Along with the work on IPFIX implementation we also work on optimization of the monitoring itself and adding some further features which are not covered by the IPFIX specification.

- Development of tools utilizing the monitoring in different application domains [21] – the main goal is the utilization of the information obtained by the BasicMeter tool for different purposes such as usage-based accounting, SLA evaluation, intrusion detection, and traffic engineering.

- Experimental deployment of the monitoring tools in a laboratory environment and real networks with the goals of testing the functionality and interoperability, performance analysis and feedback acquirement for further development.

In the following sections we provide an overview of the activities of the MONICA group starting with a brief introduction of the BasicMeter tool and following with an overview of its enhancements with an insight into select ones.

2. The BasicMeter tool

BasicMeter [8] is a tool designed for network traffic parameter monitoring. The measurement is passive and does not require generation of additional network traffic. The concept of the BasicMeter architecture is in conformity with the IPFIX requirements [17]. Its main components are depicted in Figure 1.

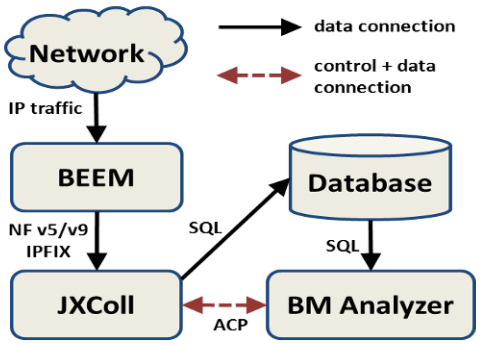

Figure 1. The architecture of the BasicMeter tool.

Their brief description is the following:

- BEEM – stands for BasicmEter Exporting and Metering process. It covers packet capturing, filtering, sampling, creating and maintaining of flow records in the flow cache; and exporting of flow records from the observation point by the IPFIX protocol [2]. It is a console application written in the C language. BEEM can be configured by the modification of its XML configuration file.

- JXColl – stands for Java XML Collector. It can collect flow records exported from several observation points in the format of protocols Netflow v5, Netflow v9 or IPFIX. Flow records can be stored in a database (for future use and analysis) and/or directly sent to an analyzing application by the ACP. JXColl can also generate accounting records and store them into a special database for usage-based accounting. JXColl can be configured by the modification of its XML configuration file.

- BM Analyzer – stands for the BasicMeter Analyzer. It provides user front-end for both the visualization of the information obtained by observation points and management of the single components of the BasicMeter tool. BM Analyzer is a standalone Java application accessible via Java Webstart technology, but the ongoing version is designed as a modular web application integrating centralized management and many potential IPFIX applicabilities [21].

- ACP — stands for Analyzer – Collector Protocol [4, 7] which serves for the direct communication of the JXColl and BM Analyzer making this communication fast enough for the real-time monitoring.

Regarding the roles of these main components (BEEM, JXColl, BM Analyzer), in the following they will be referred to as the exporter, collector and analyzer.

3. Enhancements by MONICA

As we mentioned before, along with the implementation of IPFIX based monitoring tool we work on several enhancements and add-ons too. These can be divided into two categories – enhancements of the architecture for obtaining of IP flow information, and application domains specific tools. The enhancements of the architecture for obtaining of IP flow information being developed by MONICA include:

- Analyzer – Collector Protocol (ACP) for direct communication of the collector and analyzer as a supporting mechanism for real-time monitoring [4, 7]. In the BasicMeter architecture the analyzer is intended to visualize the data computed on the basis of the gathered traffic information and manage the other components of the tool. However, the analyzer itself is not a subject of the IPFIX specifications and therefore we had to draft and implement our own network protocols for the data flow and communication between the analyzer and the other components of the metering tool. Along with the ACP we also developed an Application Programmable Interface (API), so it can be easily implemented in any analyzer-like application. More details are provided in the dedicated subsection of the paper.

- Exporter – Collector – Analyzer Manager (ECAM) for centralized management and easy deployment of the monitoring tool [11]. ECAM is implemented at each component of the BasicMeter tool providing a user-friendly configuration of all the exporters and collectors in the observation domain from the analyzer's user interface. It also provides information about the state of the exporting and the collecting process(es) and offers a possibility to control remotely the individual components of the tool by start/stop/restart commands of such processes. Meanwhile, a configuration data model has been drafted by the IPFIX WG [14], so we will follow and incorporate it into our solution.

- Data WareHouse [5] for data pre-processing and storing for efficient access by the analyzing application. This is a supporting mechanism for reducing the response time during the selection of historical flow records from the database, especially when the result of the SELECT statement is a large amount of records. More details are provided in the dedicated subsection of the paper.

- Adaptive export of flow records from the observation points for the purpose of real-time monitoring. In some application domains, where real-time monitoring is required (e.g. traffic engineering, intrusion detection, etc.), short exporting intervals should be used. However, very short intervals would lead to the exhaustion of the system resources consumed by the monitoring tool. Therefore we developed an idea of adapting the exporting interval to the changing behavior of the network traffic.

- Measurement of One-Way Delay (OWD) with compensation of the observation points' clock skew [3, 18]. The basic idea of OWD measurement seems to be very simple – subtract the timestamps for a single packet observed at different points of the measurement. However, for the implementation of this idea we have to cope with several problems such as the generation of unique packet identifiers [6], time synchronization and clocks' skew compensation of the observation points. There are already existing solutions with very high precision, but these are typically very expensive because high precision can be achieved only by using specialized hardware equipment. Our approach of OWD measurement gives reasonable precision without requiring expensive hardware. More details are provided in the dedicated subsection of the paper.

- Implementation of new information elements into the information model. Some of the information elements used in the above mentioned enhancements were not defined by the IPFIX information model [16]. Therefore we had to implement them with an Enterprise ID of '26235'.

Application domains specific tools being developed by MONICA:

- Usage-based accounting which allows more flexible accounting for network services than flat-rate accounting and brings advantages for both the service providers and users [9]. Our solution contains an accounting database and additional collector functions which create accounting records by aggregation of the flow records using specified criteria.
- SLA evaluation for tracking the fulfillment of the specified conditions between the subscriber and the provider. Our solution provides web interface for both sides with specific information about the operation and traffic parameters of the last mile. It has a modular architecture, so it is simple to add new components upon the needs of such type of users.
- Anomaly-based Intrusion Detection System (IDS) which builds information by characterizing the normal traffic, evaluates the current traffic, and alerts the administrator when violation occurs. Anomaly detection is implemented by fuzzy subsystems designated for the detection of specific types of network attacks, and also for the detection of general anomalies of the network traffic.
- Adaptive Anomaly Driven Traffic Engineering (AADTE) using different dynamically applied configuration changes into the network devices, such as routing optimization, usage of QoS mechanisms or security policy enforcement [10]. Specific optimizing changes are selected by a domain-specific decision maker module when traffic anomaly is detected.
- Monitoring of information systems where the importance of solving security issues led us to design the monitoring tool's architecture inspired by the IPFIX. The proposed monitoring tool's architecture involves the measurement of the operational parameters of an information system. It is easily scalable and powerful enough to handle a huge amount of monitored operations occuring in the case of DoS and DDoS attacks on a monitored information system.

Detailed description of all the enhancements would be out of the scope of this paper, thus in the following subsections we present just select ones.

3.1. Analyzer – Collector Protocol

In the primary concept of the BasicMeter tool, the measured data should be stored in a database for later analysis. However, this method is absolutely inapt in the case of real-time monitoring.
The time necessary for the analyzer to get the traffic information using a database depends on many factors that are difficult to estimate. These factors are, e.g., the time needed by the exporter to prepare and transmit the flow record to the collector; the time required by the collector to parse the flow record and transmit the data to the database server; or the time required by the analyzer to process the received data from the database. Another significant time period is the one that is consumed by the database server to store the data and to return the result of the analyzer's query. This time period is highly dependent on various database technologies and data storing techniques. Moreover, with the growth of data in the database, the processing of the queries is becoming more and more time-consuming.
For this reason we developed the Analyzer – Collector Protocol (ACP) which allows the analyzer to gain data about network traffic directly from the collector, without the ineffective access to the database. The collector is permanently inserting the traffic data into a database and if needed, by the means of ACP it is simultaneously sending exactly the same data — or by a customizable filter a part of it — to the analyzer.
ACP is a binary, application layer protocol. The communication is bidirectional (TCP is used for transport) and works on the basis of the client-server model, where the client side is represented by the analyzer and the server side is represented by the collector. Over ACP, besides the data, control messages are also transmitted. These messages are dedicated for format check of the received data by the analyzer and also for the control of the communication itself.
The communication is based on sending queries by the analyzer and sending replies with data by the collector. This type of communication could be easily illustrated by a Finite State Machine (FSM) of both sides [4]. The particular communication phases are represented with the machine's states at which every message has its own unique identifier.

Figures 2a and 2b depict the state transitions of the Analyzer – Collector Protocol. The types of the control messages are:

- MA – authentication;
- M0 – setting template;
- M1 – setting filter;
- M2 – suspending data transmission;
- M3 – resuming data transmission;
- M4 – setting data transmission type;
- M5 – acknowledging the received data.

The state graph in Figure 2a represents the collector's side communication phases, where for example M0_TV1/R1 means, that when the collector receives a request for setting a template (M0) and all the criteria are satisfied (TV1), it responds to the analyzer with an acceptance (R1) and traverses from state S_2 to state S_4. Control messages sent by the analyzer can be the following:

- MA_TV0 (MA_TV1) – incorrect (correct) authentication data;
- M0_TV0 (M0_TV1) – incorrect (correct) template;
- M1_TV0 (M1_TV1) – incorrect (correct) filter;
- M2_TV0 (M2_TV1) – data transmission unsuccessfully (successfully) suspended;
- M3_TV0 (M3_TV1) – data transmission unsuccessfully (successfully) resumed;
- M4_TV0 (M4_TV1) – not supported (accepted) mode of data transmission;
- M5 – acknowledging the received data.

M indicates the type of the control message, TV indicates the truth value of the request and R indicates the particular responses (rejection or acceptance) from the collector.
State graph in Figure 2b is describing the analyzer's side communication phases where according to the previous example R1 means, that when the analyzer receives an acceptance (R1) in state S_1 for its request from the collector, it traverses from state S_2 to state S_3 and starts to receive the data. The reason that the outputs are not present in this state graph is, that they represent the particular activities of the analyzer, i.e. receiving the data. Control messages sent by the collector can be the following:

- R0 (R1) – access denied (guaranteed) in state S_1;
- R0 (R1) – the requested template was rejected (accepted) in state S_2;
- R10 (R10) – filter rejected (accepted);
- R20 (R21) – suspension of data transmission was rejected (accepted);
- R30 (R31) – resume of data transmission was rejected (accepted);
- R40 (R41) – mode of data transmission is not supported (mode of data transmission supported and also accepted).

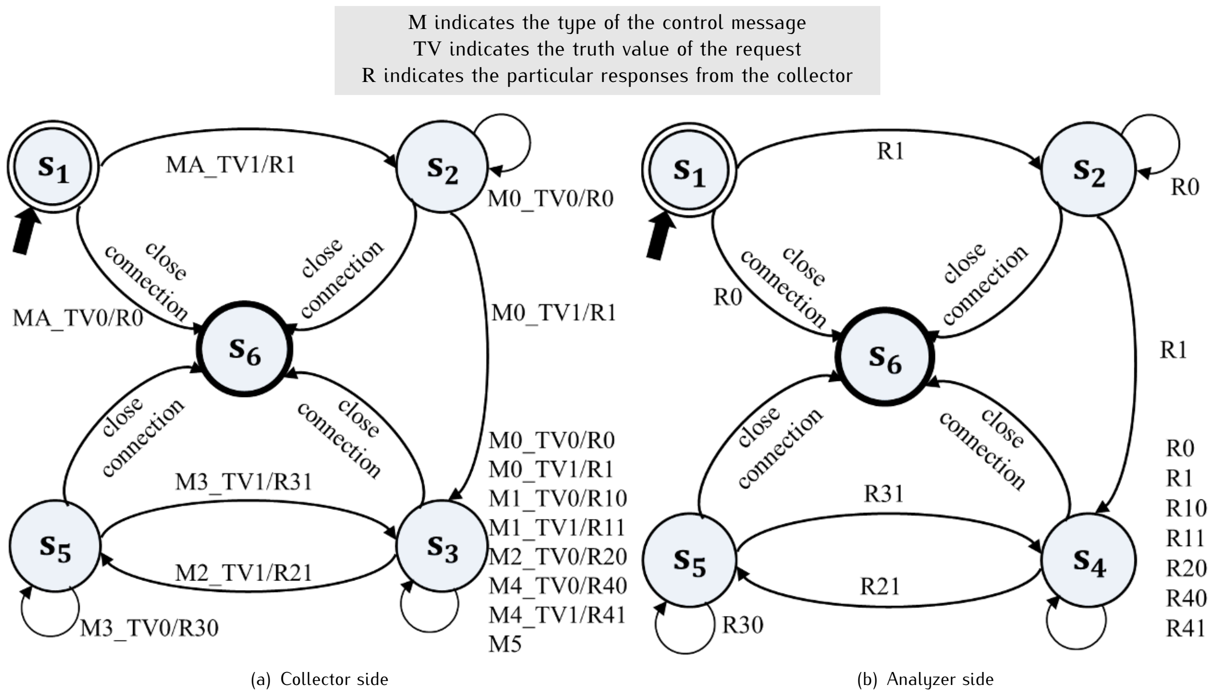

Figure 2. State transitions of the Analyzer – Collector Protocol.

At the initial state, the collector is awaiting the analyzer's connection at a pre-agreed port (2138 by default). If the connection is established successfully both sides traverse to their S_1 (Start) states. Before the bi-directional communication can be opened, the analyzer has to send authentication data (MA) to the collector. On each request the collector responses with an acceptance (TV1) or rejection (TV0). Successful authentication brings both sides to their S_2 (Waiting) states, otherwise the collector terminates the connection.
In S_2 state the analyzer using the template message (M0) can set the desired format of the received data. If the template is accepted, the collector traverses to its S_4 (Transmitting) state and starts with the data transmission in the format of the accepted template.
At the same time, the analyzer traverses to its S_3 (Receiving) state and starts to receive the data from the collector. In this phase the template can be changed anytime. Real data besides control messages are sent/received only in states S_3/S_4.
By default, the collector exports all traffic information obtained from the exporter(s). However, when necessary, it is also possible to receive only specific information by setting the filter (M1). This optional step is accepted by the collector only in its S_4 state, while only one filter can be set in a time. The filter can be canceled or replaced with an empty filter rule in the collector's S_4 state. Traffic information can be filtered on the basis of source, destination and measuring point's IP addresses; source and destination port; and protocol.
When necessary, the transmission of the data can be suspended (M2) anytime. In such a case the collector first sends the remaining data from the currently processed flow record to the analyzer and then both sides traverse to their S_5 (Suspended) states. In general, the collector has no reason to discard the suspension request. The data transmission can be resumed (M3) anytime which brings both sides to their previous states.
By default, the collector sends the information elements in N-tuples, which means that during one cycle of data reception the analyzer receives N values. This type of data transmission (M4) can be changed optionally in the collector's S_4 state. Information elements can be also transmitted one-by-one. Data transfer synchronization is ensured by acknowledging the received data (M5).
Unfortunately, receiving data by the ACP also brings a certain delay caused by the processing time in the lower part of the tool. However, this time period compared to the one when obtaining traffic data using a database is significantly less. Thus, from the point of real-time monitoring, gathering traffic data by the ACP is more affordable [7].

3.1.1. Application programmable interface for the ACP

In the BasicMeter architecture while only one collector is needed, there could be more than one analyzer (analyzers may vary in structures and purposes). In this case the ACP would need to be implemented more than once. While the implementation of this kind of protocol is in some cases really problematic, there was a need for an Application Programmable Interface (API) which would simplify the implementation process of the ACP in various analyzers.
For this reason we developed the Application Programmable Interface for the Analyzer – Collector Protocol (ACPapi). ACPapi was designed to simplify the work of the programmers during the development of real-time data dependent applications [15]. Its main purpose is to serve the ACP, whereby the programmers do not have to care for the implementation details of this protocol. Obtaining the data with ACPapi was proposed as a design pattern in which the incoming messages and data are processed in a cycle. The data could be processed by other methods too, but from the point of real-time data processing, the method when the developer is forced to analyze the received data immediately is more favorable.

3.1.2. Future work

In the current version of the ACP, the templates are sent as arrays containing the identifiers of the information elements. However, in the future we plan to replace the arrays with a common binary description of these elements. In case of ACPapi, future work should be aimed at the improvement of the actual design pattern.

3.2. Anomaly-based intrusion detection

The BasicMeter Intrusion Detection System (bmIDS) is an anomaly-based IDS, which combines network traffic metrics with fuzzy rules to determine the probability of a specific network attack [1]. bmIDS, as depicted in Figure 3, consists of the following parts:

- **bmIDS Analyzer** represents the main function of the IDS, namely traffic analysis and network attacks probability evaluation [1]. The inputs of the analyzer are information about IP flows received from the collector through the ACP protocol [4, 7]. The output of the bmIDS Analyzer is the probability of the intrusion. bmIDS Analyzer was proposed with a modular approach, where each intrusion type has its own module. So the detection of various network intrusions is executed simultaneously and independently. bmIDS Analyzer can be further divided into the following components:
  - **TrafficProcessor** is an abstract component for processing the IP flows. On the basis of attack characteristics an individual component accumulates the values of selected traffic parameters for each type of attack. At regular intervals, the processed information are also sent to another component for evaluation.
  - **FuzzyDetector** receives accumulated data from the relevant processors and evaluates it using fuzzy logic and fuzzy sets. As the previous component, it consists of several modules, each of which evaluates traffic against the relevant type of attack. Details about critical traffic are automatically saved into a database and can be displayed by the bmIDS Web component. In case of monitoring by the administrator, the detection outputs can be also directly sent to the bmIDS Web.
- **bmIDS Web** provides a graphical interface for the administrators. The detection is performed in real-time, while the outputs are presented by charts. bmIDS Web can also display information about historical attacks. This information is obtained from the database, in which it was inserted by the bmIDS Analyzer.

The bmIDS can be used in two modes:

- **Learning mode** was designed to provide standard traffic profiles for specific parameters. The profiles are created from historical records about flows stored in the database. These profiles are then used to create fuzzy sets for individual fuzzy subsystems.
- **Detection mode** performs the main function, which is real-time network attack detection. bmIDS in its present state can detect three types of network attacks:

- **SYN Flood** – for the detection, the actual synCount (number of TCP SYN packets and with the same destination IP address and port number) and past synCount (synCount in two previous cycles of evaluation) are used as input variables.
- **UDP Flood** – for the detection, the actual udpPacketCount (number of UDP packets with the same destination IP address and random port number) and past udpPacketCount (udpPacketCount in two previous cycles of evaluation) are used as input variables.
- **Port Scan** – for the detection, the flowCount (number of IP flows with the same IP addresses and different destination port number) is used as the input variable.

SYN Flood and UDP Flood attacks usually have to be performed for a longer time interval to cause damages to the target system. Therefore their detectors work with two input variables, the actual and past values of its characteristic. However, Port Scan attack uses only one variable, which in the actual cycle of evaluation (repeated every 5 seconds) is the flowCount variable. All these characteristics are obtained from the accumulated IP flows.

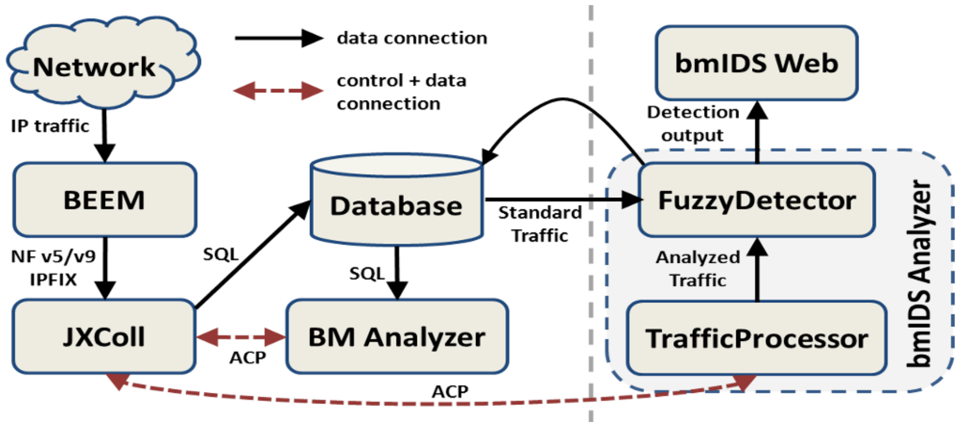

Figure 3. The architecture of the BasicMeter Intrusion Detection System.

3.2.1. *Future work*

Our future work will be aimed at expanding the bmIDS with the detection of other known or unknown network intrusion types.

3.3. Measurement of OWD with compensation of observation points' clock skew

One of the most significant parameters of network traffic is the delay. Among different types of delay, One-Way Delay (OWD) is one of the most important, however its determination is also most difficult. The measurement of OWD is based on the principle of sending a message with a timestamp between two endpoints, where the time of sending and receiving of this message is recorded. The value of OWD is given by the difference of the times, which is correct only in the case when the clocks on both endpoints are synchronized [3, 12]. Unfortunately this condition is not easy to fulfill. Moreover, the principle of OWD measurement would require generation of additional traffic for the timestamps. Unfortunately this would violate the passive measurements concept of the Basicmeter tool. To cope with these problems we designed and implemented the measurement of OWD in the following way.
In the architecture of the BasicMeter tool, the individual exporters are sending flow records describing the captured traffic to the collector. This allowed us to carry timestamps from the two endpoints (at least two exporters are needed) to the collector and calculate the delay directly in it. Moreover, the individual flow records include various information, i.e. time characteristics and flow identifiers. So, instead of sending the timestamps in dedicated messages, we used IPFIX information elements describing the time characteristics (flowStart and flowEnd).
To calculate OWD we needed information about such a packet, which passes the two metering points where flow characteristics are recorded. Such packets are the first and last packets of the flow. If the absolute start and end time of

the flow is not identical, we get two timestamps of two IP packets, otherwise we get only one timestamp of one IP packet. Based on the count of acquired timestamps we are able to calculate one or two OWD values per flow. Unfortunately it is uncertain whether the IP packet captured within a flow by one metering point is identical to the IP packet captured within the same flow by another metering point. In this case we would get inaccurate OWD values. To avoid such a situation we used a method described in [6]. With this method, on the basis of certain fields of IP packet and data transmitted in it, is possible to generate an identifier for the first and last IP packets of the flow. The selected fields of the IP packet header are as follows:

- Header length,
- Total length,
- Identification,
- Protocol,
- Source address,
- Destination address.

The method was tested on over 350 000 packets. There were 3.81% of non-unique identifiers from which 2.64% packets were absolutely identical, so there was no chance to generate unique identifiers. All this resulted in the fact that only 1.17% of the identifiers were generated by the fault of the proposed method [6].
For the transmission of these identifiers from exporter(s) to collector we created two new information elements (firstPacketID and lastPacketID) which along with the other information are exported in the flow records. The definition of these information elements are described in Table 1.

Table 1. The list of IFPIX information elements created for carrying the identifiers of the first and last IP packet of the flow.

ID	Enterprise ID	Name	Type
242	26 235	firstPacketID	octetArray(16)
243	26 235	lastPacketID	octetArray(16)

Based on these facts, the absolute value of the difference of two corresponding timestamps will indicate the value of OWD.
Corresponding timestamps of flow records are those ones that satisfy the following conditions [3]:

- both flow records came from different metering points,
- both have absolute times of the start (flowStart) or end (flowEnd) of the flow,
- both have identical identifiers of the first (firstPacketID) or last (lastPacketID) IP packet of the flow,
- both belong to the same flow.

However, before we could make the actual calculation of One-Way Delay, we had to ensure the above mentioned metering points' clocks synchronization too. Unfortunately neither of the existing methods for clock synchronization [18] have met our needs. Therefore we designed our own method for metering points' clocks synchronization.
If we want to synchronize two clocks, first we have to find out how they differ [13]. So we have performed a measurement. We were sending timestamp t_1 with actual time from metering point with clock C_s (sender clock) to metering point with clock C_r (receiver clock). Timestamp t_2 was set with the actual time of arrival of timestamp t_1 in the metering point with clock C_r. The subtraction of the two times $\Delta t = (t_2 - t_1)$ gave the offset between clocks C_s and C_r. We have noticed, that Δt values represent a constantly increasing $y = a.x + b$ linear function, as depicted in Figure 4 (data set represented with upper line). This linear function describes variable clock offset between C_s and C_r at time t.

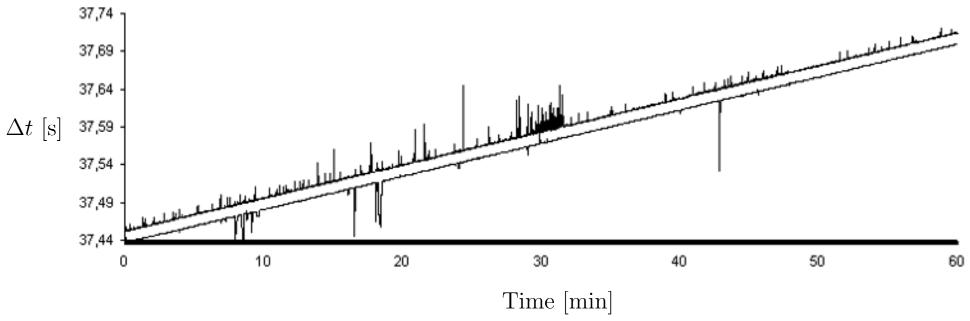

Figure 4. Measured offset between two clocks containing one-way delay.

If we want to determine the parameters of the linear function, first we have to obtain the changing value of the offset. The problem is, that the time difference Δt is not the actual clock offset, but it is combined with the delay of the direct connection of the two network endpoints. So we need to measure One-Way Delay between those endpoints and subtract it from Δt. The result of this subtraction is depicted in Figure 4 by the data set represented by the bottom line. Except the synchronization is also important to compensate clocks' skew delay (even in the case of using high precision synchronization method, e.g. GPS). For this purpose, we drafted a new way of clock offset measurement and clock skew delay compensation.

In the BasicMeter architecture one collector is superior to a number of exporters. This hierarchy led us to choose the collector as the reference point of the synchronization. So in our method of synchronization the metering points are synchronized towards the collector. For synchronization purpose the exporter is sending a timestamp t_0 containing the time of the transmission of the stamp and receiving a timestamp t_1 containing the time of the reception of stamp t_0 in the collector. Then the exporter creates a timestamp t_2 containing the time of the reception of stamp t_1 from the collector. Based on these three times, One-Way Delay $OWD = (t_2 - t_0)/2$; and real clock offset $o = (t_2 - t_1) - OWD$ can be calculated.

After the approximation of the clock offset with a linear function, we got two parameters a and b. The clock correction for the metering points then could be easily calculated as follows:

$$C_s^s(t) = C_s(t) - (a.t + b),$$

where C_s^s means a synchronized clock on metering point's side with clock C_s, and a and b are the parameters acquired via the approximation.

When all previously mentioned conditions are satisfied the collector obtains two corresponding timestamps belonging to the same IP packet, which has passed both synchronized metering points. In this case OWD is calculated by the collector and sent to the database. Otherwise the collector holds the timestamps in a buffer for a customizable amount of time. If during this time no more corresponding timestamps can be found the calculation of OWD will be not performed and the timestamps will be removed from the buffer.

The experiments proved that with the above described method for OWD measurement one could calculate accurate values with a precision of microseconds [18].

3.3.1. Future work

Our future work will focus on the solution of the issues related to the synchronization convergence in extensive measuring networks with a large number of metering points. In addition, the problem of measuring link asymmetry will be a subject of further research by the authors. Also factors influencing instability of the inaccuracy rate will be researched, which is needed for the determination of appropriate period for its recalculation. Finally, our future work will also take into account the emerging technologies for precise time synchronization between network nodes, such as IEEE 1588 PTP or ITU-T G.6282/Y.1362 Synchronous Ethernet.

3.4. Data WareHouse for data preprocessing and storing for efficient access by the analyzing application

Processing the analyzers' queries by the database is very time-consuming even in the case of small amount of stored data. With the growth of data, the time required by the database to process the queries can exceed more than tens of seconds. Further problems occur in the analyzer where it is difficult to implement the effective evaluation of the graphical visualization of a huge amount of data (millions of records) obtained from the database. Moreover, it is assumed that the analyzer will be able to create a chart from a much larger time interval (records). Unfortunately, this problem had an adverse impact on the functionality of the BasicMeter tool. After considering the factors described in [20], the Data Warehouse technology with some other optimization improvements were implemented in the database of the BasicMeter tool.

First, a new database was created. The principle of filling the tables of this database was fully inspired by the Data Warehouse technology, where the records are representing a kind of a preprocessed and summarized – to a precision of one minute of time unit – data of selected information elements. Using the PL/pgSQL language, the following database objects[1] were created in the database of the BasicMeter tool:

- function (stored procedure),
- trigger pertinent to this function.

After each insertion of a record into the database, the trigger calls its pertaining function. Subsequently, the function starts to insert the records, obtained on the basis of the actually stored values in the default database, into the new database in the following order:

1. First the command by which the function was called (it has to be an INSERT statement) and the end of the flow identifier is checked (examining the end of the flow identifier is needed to reduce duplicate records). When these criteria are satisfied, the function connects to the new database, where the proposed tables for efficient querying were created.
2. In the next step the function begins to store the records in the tables. First it checks whether the entry already exists in the table proposed for unique records. If the record is already there, only its duplicate record counter field will be increased. Otherwise the new, unique entry will be stored in the table.
3. In case that the actually inserted records meets the criteria of some distinguished information, this record will be also stored in the tables dedicated for preprocessing and summarizing purposes.
4. In the last step the preprocessed data about the start and end of the flow and some other entries characterizing the IP flow will be stored in the tables.

Next, when new data were inserted in the old database, preprocessed and summarized data were also inserted into the new database. This made less demanding the acquirement and visualization of the data by the analyzer, which resulted in significant reduction of the time required by the database for processing the query and providing the results to the analyzer and by the analyzer to visualize the acquired data.

3.4.1. Verifying the benefits of the enhancements

The verification of the expected results was performed on a topology as illustrated in Figure 5. The verification of the proposed tables for preprocessed and summarized records was performed by measurement, which was repeated 20 times. Approximately 2 million records were collected by the BasicMeter tool. During the monitoring, the measured network traffic information was sent by the exporter to the collector (continuous line in Figure 5), where after processing the IPFIX flow records, the obtained data were stored in the database (fine dashed line in Figure 5).

The comparison of the average times before and after the transition to the Data Warehouse brought the following results. Before the optimization, it took the BasicMeter approximately 20 seconds to visualize the information from about 2 million

[1] *The PostgreSQL Global Developement Group, PostgreSQL Documentation, 2011*

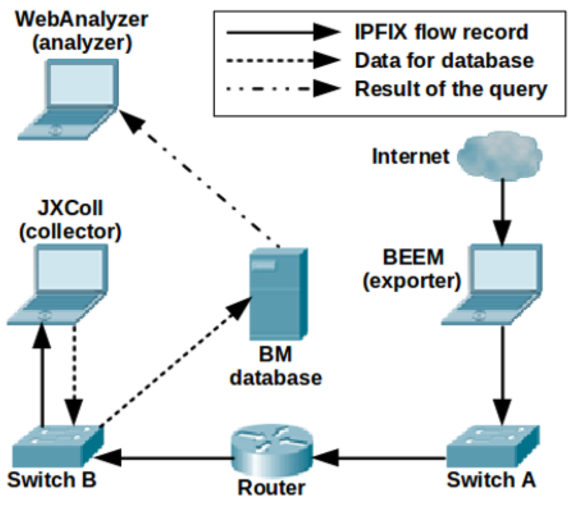

Figure 5. The topology of the verification process.

data records. After implementing the Data Warehouse technology the analyzer did not have to create the chart from 2 million data records, but on the basis of the time characteristics it made do with only the preprocessed data. So the evaluation and processing time was successfully reduced to a half second and the visualization was performed in each case of the verification measurement without any noticeable delay. However the precision of the visualization through the proposed optimization steps is only one minute of time unit, for more precise visualization the default database could be used.

3.4.2. Future work

Although the optimization steps over the database succeeded, they serve only as a temporary solution because not all requirements during the implementation of Data Warehouse technology were completely defined. Therefore, in the future the database of the BasicMeter should be first subjected to thorough analysis, during which all the requirements from both the collector and analyzer should be definitely identified. Consequently the preprocessing and aggregation of data on the basis of time characteristics should be redesigned and implemented.
Future work should be also aimed at the way of filling the new database containing unique and summarized records. The preferable way would be to calculate and store these records at a time (during weekend or at midnight) that has the least impact on the performance of the network monitoring.

4. Conclusion

The goal of this paper was to introduce the MONICA research group which is focused on network monitoring and related areas. We provided an overview of the MONICA activities and insight into select ones, while pointing out some of the future work as well. We are open to the discussion and any kind of cooperation in this area – further development of above mentioned or other related topics, co-publication of ideas and results, or cooperation in drafting of new parts of the IPFIX architecture.

Acknowledgments

This work is the result of the project implementation: Development of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

References

[1] Chandola V., Banerjee A., Kumar V., Anomaly Detection - A Survey, ACM Comput. Surv., 41, 15:1-15:58, 2009
[2] Claise B., Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, RFC, 5101, 2008
[3] Giertl J., Husivarga Ľ., Révés M., Pekár A., Feciľak P., Measurement of Network Traffic Time Parameters, In: International Scientific Conference on Informatics (INFORMATICS) (November 2011, Rožňava, SK), 33-37, 2011
[4] Giertl J., Jakab R., Koščo Ľ., Communication Protocol in Computer Network Performance Parameters Measurement, In: International Information and Telecommunication Technologies Symposium (I2TS) (December 2005, Florianopolis, Brazil), 161-162, 2005
[5] Inmon W. H., Building the Data Warehouse (Fourth Edition, Wiley Publishing, Inc., Indianapolis, IN, USA), 2005
[6] Jakab F., Baldovský G., Genči J., Giertl J., Unique Packet Identifiers for Multipoint Monitoring of QoS Parameters, In: International Scientific Conference on Electronic Computers and Informatics (ECI) (September 2006, Herľany, SK), 249-254, 2006
[7] Jakab F., Giertl J., Jakab R., Kaščák M., Improving Efficiency and Manageability in IPFIX Network Monitoring Platform, In: International Network Conference (INC) (July 2006, Plymouth, UK), 81-88, 2006
[8] Jakab F., Koščo Ľ., Potocký M., Giertl J., Contribution to QoS Parameters Measurement: The BasicMeter Project, In: International Conference on Emerging e-learning Technologies and Applications (ICETA), 371-377, 2005
[9] Kaščák M., Jakab F., Giertl J., Révés M., Usage-based Accounting, Comput. Sci. Technol. Res. Surv., 5, 33-38, 2010
[10] Kopka J., Révés M., Giertl J., Anomaly Detection Techniques for Adaptive Anomaly Driven Traffic Engineering, In: Scientific Conference of Young Researchers (SCYR) (May 2010, Košice, SK), 254-257, 2010
[11] Mihok T., Giertl J., Révés M., Pekár A., Feciľak P., System for Centralized Management of the BasicMeter Tool, IP Networking I - Theory and Practice, 2011
[12] Mills D., Modelling and Analysis of Computer Network Clocks, In: University of Delaware - Electrical Engineering Department Report, 92-5-2, 29, 1992
[13] Moon S.B., Skelly P., Towsley D., Estimation and Removal of Clock Skew from Network Delay Measurements, In: IEEE International Conference on Computer Communications (IEEE INFOCOM) (March 1999, Amherst, MA, USA), 227-234, 1999
[14] Muenz G., Claise B., Aitken P., Configuration Data Model for IPFIX and PSAMP, Internet-Draft, 2011
[15] Pekár A., Giertl J., Révés M., Feciľak P., Simplification of the Real-Time Network Traffic Monitoring, In: Proceeding of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice – Electrical Engineering and Informatics II, EEI II, 272-277, 2011
[16] Quittek J., Bryant S., Claise B., Aitken P., Meyer J., Information Model for IP Flow Information Export, RFC, 5102, 2008
[17] Quittek J., Zseby T., Claise B., Zander S., Requirements for IP Flow Information Export (IPFIX), RFC, 3917, 2004
[18] Révés M., Giertl J., Jakab F., Improvement of Synchronization Accuracy in the Monitoring of Computer Networks, MJCN, 4, 37-43, 2008
[19] Sadasivan G., Brownlee N., Claise B., Quittek J., Architecture for IP Flow Information Export, RFC, 5470, 2009
[20] Vokorokos L., Pekár A., Ádám N., Preparing Databases for Network Traffic Monitoring, International Symposium on Applied Machine Intelligence and Informatics (SAMI) (January 2012, Herľany, SK), 12-18, 2012
[21] Zseby T., Boschi E., Brownlee N., Claise B., IP Flow Information Export (IPFIX) Applicability, RFC, 5472, 2009
[22] Zseby T., Molina M., Duffield N., Niccolini S., Raspall F., Sampling and Filtering Techniques for IP Packet Selection, RFC, 5475, 2008