A Framework for Classifying IPFIX Flow Data, Case KNN Classifier

Jussi Nieminen, Jorma Ylinen, Timo Seppälä, Teemu Alapaholuoma, Pekka Loula
Telecommunication Research Center
Tampere University of Technology, Pori Unit
Pori, Finland
jussi.nieminen@tut.fi, jorma.ylinen@tut.fi, timo.a.seppala@tut.fi, teemu.alapaholuoma@tut.fi, pekka.loula@tut.fi

***Abstract* — Flow-level measurement applications and analysis in IP networks are inevitably gaining popularity, due to the unstoppable increase in the amount of transmitted data on the Internet. It is not reasonable or even possible to examine each and every packet traversing through a network. Our research focuses on passive flow level data classification and characteristic identification. To be more exact, our goal is to design a framework for extracting certain classes, feature(s) and behavior from IP flow data. One of the goals is to achieve this without examining the payload of any of the IP packets and without compromising the anonymity of the flow counterparts. Traditionally, Deep Packet Inspection or port mapping techniques have been applied for this purpose. In this paper, we present an alternative framework for classifying the IP traffic, which we aim to utilize in the future for separating classes from the IP traffic for information security purposes.**

Keywords-Flow; IP; IPFIX; KNN; Classification

I. INTRODUCTION

In this paper, we study the possibility of identifying traffic characteristics from IP traffic, and more precisely from the IP/TCP/UDP/ICMP header data. We utilize the KNN Classifier method (K Nearest Neighbors) through passive data analysis on IPFIX [1] [2] flow data. The motivation for our research comes from the area of information security. We are keen on finding methods for separating classes from the data in order to be able to identify a measurable unit (IPFIX flow in this case) for example as normal or malicious in future analysis work. In this paper, we present a framework, which can be utilized for that purpose. Our research relies on total anonymity. The IP-addresses are either anonymized or cut off prior to analysis execution. The payload of each IP packet is cut off in the data capture phase, so all the details compromising the user privacy of the connection counterparts are discarded.

The KNN Classifier method determines the class of a new data point based on its K-nearest neighbors in a selected feature space. The class that exists the most among the K-nearest neighbors is given to the test data point. The KNN Classifier is based simply on the distance metric of data points. The Euclidean distance metric is the most common one, while also other metric methods are available. This obviously means that a variety of different KNN implementations have been introduced.

Our data for the analysis was captured from a large-scale local area network. The selected network is known to have a large amount of hosts and good set of services active. It is also known that the information security policy doesn't restrict the usage of any service in the network. This is a clear advantage from the analysis point of view, because the captured data is as pure as it can be without any restrictions or filtering in any way at any point.

The data was captured from the network and stored to disk in IPFIX format. In the analysis phase the data was first divided into two classes. We use a class distribution of WWW-type traffic versus other traffic in this paper. WWW as a service provides interesting viewpoints for future analysis, as it is commonly used, uses standard port numbers, and therefore also has a lot of information security aspects. The following step was to select the parameters for the classification execution. K-fold cross-validation was used as the classification framework to determine the best value for the constant 'K' in KNN-Classifier. Another important factor was to select suitable input parameters (features) for the classification. We came up with a set of three parameters. Once the parameters were selected, the actual classification was executed. As a result, the details were obtained about how the classification succeeded. The results were studied and written down, along with conclusions and observations about the functionality of the analysis framework and the methods used. Based on the analysis, we present our framework for classifying IP Flow data. In addition, some thoughts on how the results could be utilized in practice are provided.

This paper consists of seven sections. In the next section, the related work in the field of IP-traffic data classification is presented and analyzed briefly. In Section three, the data is presented in terms of how the data is obtained, how it is pre-processed, what is the total amount of data and how it is connected to real life time-wise. The theory behind the analysis is presented in Section four. Section five presents the analysis framework and the execution of each step during the analysis. The observations and results of the analysis are presented in Section six. Finally, conclusions and future plans are given in Section seven.

II. RELATED WORK

The quest for finding solutions for extracting IP-traffic characteristics from IP traffic has been a challenge for

 ISBN: 978-1-61208-186-1

researchers since the early years of the Internet. Words like generic, dynamic, effective, intelligent and self-learning are all features of a desirable solution. DPI (Deep Packet Inspection) techniques have been found effective to a certain degree by several studies. The drawback of DPI techniques is that you have to examine the payload of each and every IP packet, which is very expensive from the resource usage point of view in large-scale IP networks. Payload inspection might also compromise the anonymity of the connection counterparts, which might be unacceptable in some cases. Bendrath has examined the effects of DPI from the Internet governance point of view very carefully in his research [3]. Along with DPI techniques a variety of transport port-based methods have been introduced. These methods rely on a static port number mapping, where a certain port number is linked to a certain service in the network. Direct mapping is obviously effective but somewhat unreliable due to the possibility of port number faking or misuse. For example, Karagiannis et al. have made similar observations in their research work [4].

Various classification and clustering methods for grouping IP traffic have been introduced over the years. The focus is typically similar to this paper. By defining an analysis framework and utilizing a chosen method, a solution to a given problem is presented. Kumpulainen et al. have successfully utilized multi-level K-means clustering for separating traffic classes and behavioral patterns from IP-traffic [5]. Karagiannis et al. have used their own approach by classifying IP traffic in a three-layer classification setup. Their framework classifies the data in social, functional and application levels [4].

Moore et al. have successfully utilized a Naïve-Bayes classifier for identifying application details from network traffic. They achieved a significant improvement to the classification result by training the classifier with several simple operations [6].

In their paper, Yarifard et al. study unsupervised learning methods for identifying application specific behavior patterns from IPFIX flow data. They studied three different clustering algorithms and got good results from K-means and SNN-clustering [7].

Nguyen et al. examined a vast variety of machine learning techniques for classifying internet traffic in their paper [8]. This is an informational study rather than a survey focusing on mining the data with different methods. It provides a good overview of the methods studied and their benefits and drawbacks.

Countless studies with different goals and problem settings are available. Anyhow, there are not many papers focusing on IPFIX flow data classification. Furthermore, the use of K-Nearest-Neighbor Classification algorithm is rare in the area of IPFIX flow data classification. Based on the work of other researchers, and by our previous experience on IP traffic analysis, we decided to present our framework for IP traffic classification purposes.

III. THE DATA

The analyzed dataset was generated from a three-day trace taken in April 2011. The tracing was executed over a period of three weekdays from Tuesday to Thursday. The monitored network can be considered as a Wide Area Network (WAN) or a large-scale local area network. We use the latter term in this paper. The target network is ideal for capturing IP traffic for analysis purposes, because the information security policy of the administrating organization allows the use of any service as long as it is not illegal, does not violate the user privacy or disturb other users of the network.

The IPFIX format was used for the flow data. The IPFIX format was selected because it is the leading flow standard at the moment in terms of the level of standardization. IPFIX is based on Netflow [9], a trademark of Cisco Company. The IPFIX flow data was generated with the Maji program, provided by the WAND research team from the University of Waikato in New Zealand [10]. Maji relies heavily on the libtrace data capture library [11], which clearly also played a very important role during the data capture phase. Libtrace was also provided by the WAND -research team. Maji supports a variety of IPFIX-compliant parameters. From these parameters, we gathered a compact set of variables suitable for our purposes in order to avoid unnecessary load during the capture phase and in order to optimize the usage of storage space. The IPFIX flow data was first stored to hard disk in SQLite database format [12], from which we were able to post-process the data to CSV format for the analysis execution.

We further reduced the dataset to include only needed parameters. The dataset ended up holding in a 123 million rows, i.e., IPFIX flow records. For each flow record there is a set of parameters as follows:

1. Feature identifier (WWW-type or Other)
2. Source Transport Port
3. Destination Transport Port
4. Number of transmitted packets within the flow
5. Number of transmitted octets within the flow
6. Maximum Time To Live value within the flow

Parameters one, two, and three are related to the class definition. We separated the traffic that looks WWW-related from the rest of the data. We call this phase the basic profiling phase. The purpose of basic profiling is to highlight the desired feature or traffic class. After basic profiling we should be able to trust that the profiled traffic is what it looks like with acceptable probability. Parameters four, five and six were chosen as input parameters for the actual classification phase. They were selected after some preliminary testing and visual data mining of the flow parameters. As a guideline for parameter selection, we used the characteristics of the KNN Classifier, meaning that we tried to find parameters whose value distribution was somehow clustered or packed into clear groups within the value range. The better these conditions are met, the better is the probability of finding the correct class for a test

 ISBN: 978-1-61208-186-1

observation. The parameter selection also involved a behavioral factor. We went through the flow parameters and wrote down characteristics typical for traffic that looks like WWW traffic and then used those facts in the selection.

IV. THE THEORY

KNN classification belongs to the supervised learning methods in the field of machine learning techniques. Furthermore, KNN classification is a non-parametric learning method, meaning that it does not assume any known prior distribution. Naïve-Bayes for example assumes that the data follows normal distribution. Non-parametric methods are sometimes referred to as instance-based or memory-based methods.

KNN-Classifier is a simple, yet computationally expensive classification method. It is based on the distance metric of the classification features. The classifier algorithm is given a feature vector as an input and it places it in the feature space of the training dataset for comparison. Based on the constant 'K' and the selected distance metric, the algorithm computes the class for the new data point based on which class exists the most within the K nearest neighbors of the test feature vector.

KNN requires the whole training dataset to be available whenever a new test data point is set under classification. The classifier computes the distance of each test data point to each and every data point in the training dataset. This limits the use of KNN Classifier to being suitable mainly for passive data analysis rather than real-time applications.

The mathematics behind KNN Classifier is very simple. We have to compute each feature vector in the test data in order to define its location in the test feature space. Then each feature vector in the training data space is computed to define its location. Only after theses operations can we compare the locations of the test feature vector against the feature vectors inside the K neighbors in the training feature space. On the basis of that comparison we obtain the class for the test feature vector. Details about the mathematics are available in references [13] and [14].

There are four major questions one must ask himself/herself when designing a KNN-Classifier:

1. What is the characteristic in our dataset that defines the class distribution, and how should it be obtained, if not natively present?
2. What is the optimal value for the neighbor constant 'K', and how should it be obtained?
3. What distance metric should we use with this particular dataset?
4. What are the features in our dataset we need in order to be able to classify each test sample with the best possible accuracy and without redundancy?

Once these questions are answered, the rest is a straightforward matter of executing of the classification. Our framework binds together the workflow from the data capture and pre-processing to the result analysis.

V. THE EXECUTION

The execution stage defines the analysis framework and the workflow of the analysis process. The framework consists of six phases.

First, the data is captured from the target network. We are focusing mainly on the analysis methods, so this phase is not described in detail here.

The second phase involves parameter reduction, which means the removal of unnecessary flow parameters from the data. Parameters such as IP-addresses (anonymized) and timestamps are not needed in the classification phase, but are essential when the flow record is generated in the capture phase.

In third phase, the desired class distribution for the dataset is generated, if not natively present in the data. This phase is called the basic profiling phase. The purpose of this step is to ensure that each flow record belongs explicitly to one and only one class. We generated two classes: 'WWW-type' and 'Other' by using the known transport port numbers 80, 8000 and 8080.

The fourth phase deals with the classifier training, i.e., configuring the classifier. The first step of the classifier training consists of selecting the classification features. In the second step of training the distance metric for the classification was selected. We decided to use the Euclidean distance metric as it is by far the most common metric method used in data analysis in general. As the third step of the training, the KNN algorithm requires the neighbor constant 'K'. To determine the best value for 'K' we executed KNN Classifier with K-values 1-10 in a 10-fold cross-validation setup. We took a sample data of IPFIX flow data and divided it into 10 subsets of equal size. Each subset in turn acts as a test data and the other 9 subsets are combined to act as training data. All in all 100 separate classification executions are obtained, one for each combination of tested values of 'K' versus each possible cross-validation setup. The K-value with the best average classification success ratio should be selected for the actual analysis phase.

The fifth phase is the execution of the actual classification with the full dataset, and the final phase of the analysis is to analyze the results and make observations and conclusions. The analysis framework and workflow is described in Figure 1.

 ISBN: 978-1-61208-186-1

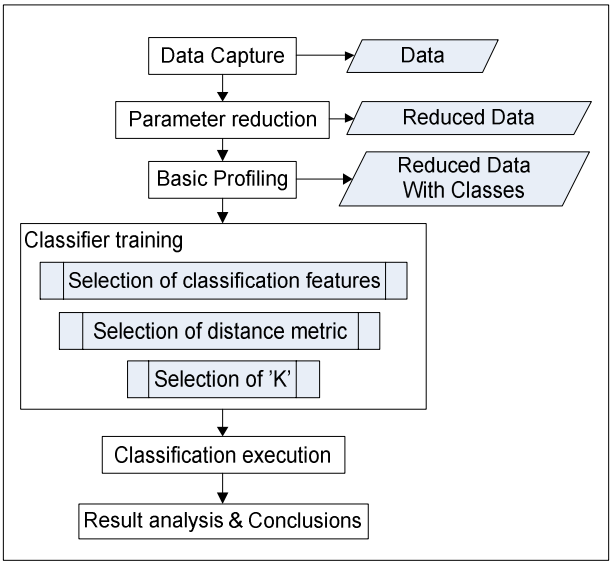

Figure 1: Analysis framework & workflow

In the actual classification phase, the three-day dataset was split into three separate datasets, as presented in Figure 2.

As we knew the week-day of each sample, we decided to split the dataset day-wise (N=3) instead of splitting the dataset into subsets of equal size. This meant that we could compare possible similarities and differences in the classification ratio between the days. A test data to Training data ratio of 1 to 9 (n=10) was used in each execution.

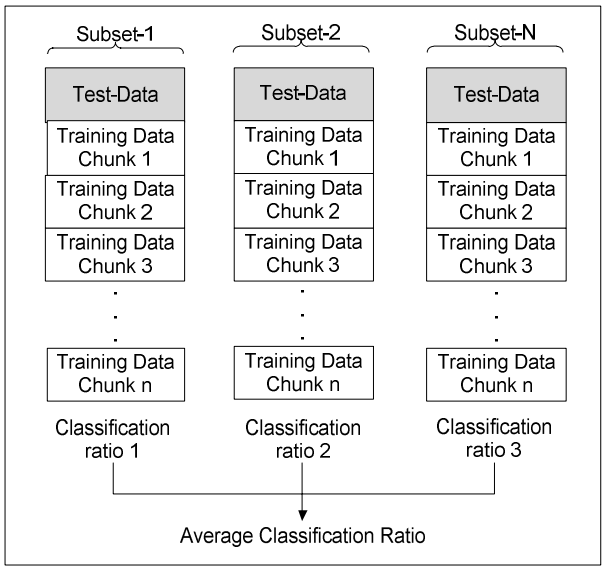

Figure 2: Test Data vs. Training Data setup

The arrangement in Figure 2 was used for two major reasons: the dataset size and to minimize the possible behavioral factor related to a certain day in the dataset. The total average classification ratio over the three-day daytime datasets was calculated in order to lighten the load and resource consumption of the classification execution. The behavioral-based division of the data derives from the fact that the amount of traffic and variety of services used in the network might be dependent on the day of the week.

KNN Classification was executed using the Euclidean distance metric. It is a straightforward and fair method for ranking observations. Moreover, the data in hand does not have any special characteristics that would require the use of more complex distance metrics. Distance-based weighting was not applied in this paper as a classifier training method.

VI. THE RESULTS

The results are handled in four parts. First we discuss the pre-processing of the dataset and the results obtained from the basic profiling phase. Then we handle the classification input parameter selection process. Subsequently, we go through the results of the actual classification. Finally, we discuss the functionality of the framework as a whole. In conclusion, we should have a view of how the applied classification mechanism and the framework in general suit the classifying of IPFIX flow data and distinguishing the feature vs. parameter relations in IPFIX flow data.

The basic profiling phase gave us a dataset with a class distribution of two classes: WWW-type and Other. We have not used the class name WWW, because we believe we can never achieve 100% success ratio in the basic profiling phase. There is always a room for error, such as measurement errors for example. However, in the case of behavior like WWW type behavior, we can be sure with an acceptable probability that the majority of the traffic traversing through ports 80, 8000 and 8080 is WWW-related. For comparison we could take DNS traffic for example. DNS is a service, which is tightly associated with port number 53. Furthermore the DNS query is static in terms of packet and flow record structure. These types of services are easier to identify in basic profiling and also easier to classify with the aid of flow features because the basis for class distribution is sufficiently solid. In this paper, the data to be classified was distributed to the aforementioned classes as follows:

TABLE I. DAILY FLOW COUNT AND CLASS DISTRIBUTION DETAILS

	Day 1	Day 2	Day 3
Flow count	41 751 116	40 350 846	40 946 094
WWW-type	12.01 %	12.27 %	11.39 %
Other	88.99 %	87.73 %	88.61 %

The data distribution was surprisingly even between the daily datasets. The traffic profile of the monitored network is very constant, at least where WWW-type traffic is concerned. The amount of WWW-type traffic was around 12% over the whole three-day dataset.

Classification input parameter selection was done by executing the classifier under several different setups and within several iterations. The goal was to train the

framework to be as generally applicable as possible for the analysis of IPFIX flow data.

In the selection of classification features, we aimed to find flow parameters that were descriptive from the client-server type of services aspect such as WWW. Another aim was to restrict the number of parameters. Our goal was to have 2 or 3 parameters to continue with. It is a clear benefit if the classification feature space has no more than three dimensions. For example, illustrating the feature space and the classification results is much easier that way. The third objective is a general goal for the classification parameter selection. The parameters should have as little redundancy as possible. Redundant parameters do not bring any distinguishable or useful information to the feature space. It is not sensible to use for example three redundant parameters if one parameter provides the same information for the classification execution. We observed that the data profiled as WWW-type consisted either of flows with a very small amount of transmitted packets or flows with a large amount of sent packets. It seemed to be a typical behavioral pattern for this type of traffic, so we decided to select the count of sent packets in the flow as one parameter. One would assume that the amount of sent packets within a flow would follow the same pattern, but in this case it did not. We decided to add it to our classification feature space as another parameter characterizing the flow without redundancy. The maximum TTL parameter was selected because it seemed to have a clear distribution into separate groups within its value range and because it was not redundant regarding the other two selected parameters.

The distance metric selection did not involve any data mining or any other characteristic examination of the data. No weighting algorithms were used either when determining the distances of the data points. The eEclidean distance metric is a clear and simple method for calculating distances between data points. Furthermore, it adds extra value to the illustration of the data since the data points can be presented and compared as a vector in a three-dimensional space.

The cross validation for examining the best value for the 'K' gave us surprisingly good results with all the tested K-values. The average classification success ratio was over 97 % and within 0.5 percent with K-values of 3-10. As a guideline, low odd values should be preferred. The best classification success ratio was achieved with a K-value of 9, both in scaled and unscaled feature space. As the difference in success ratio was very small we faced the problem of whether to go on with value 9 or to select a smaller odd value like 3 or 5 for the actual classification, which had also had a very good classification ratio throughout the cross-validation execution. Higher K-values lead to a more noise-tolerant system, but on the other hand it makes the class distribution less distinct within the k data points. Figure 3 illustrates the average classification success ratio both in un-scaled and logarithmic-scaled feature space with K-values from 1 to 10. The effect of scaling on the classification success ratio was very low. On average the success ratio increased by only about 0.2 % compared to the unscaled feature space. Our interpretation of this phenomena is that although the value ranges of Packet Count and Byte Count features are higher compared to the Maximum TTL feature, most of their data is located in the lower part of the value range, as are the values of Maximum TTL values. Therefore the effect of the logarithmic scaling has only a minor effect on the classification results.

Figure 3: Average classification ratio values from the Cross-Validation, Logarithmic scaling versus unscaled feature space, K = [1,10]

We decided to use the K-value 5 for the actual classification. None of the tested K-values in the cross-validation provided significantly better results than the others, and low odd values are typically recommended.

Several interesting pieces of information were obtained from the actual classification phase:

1. The average classification success ratio over the three classifications executed for the daily subsets.
2. The classification success ratio from each of the daily classification executions
3. The ratio of unsuccessful classifications per original class, i.e., how many 'WWW-type' flows were classified as 'Other' and vice versa.

The average classification ratio tells us the overall performance of the classification framework. The average classification ratio was 93,02 %, as shown in Table 2. This result is very good and the daily classification ratios are also very close to each other. This means that the similarity level of the daily subsets is high and the level of activity and WWW-type traffic behavior in this particular network is close to the same on different days of the week. Here we have one example of how this framework can be utilized, as a method for finding out the overall behavior of the dataset.

TABLE II. DAILY AND AVERAGE CLASSIFICATION SUCCES RATIOS

	Success ratio
Day 1	92.91 %
Day 2	91.76 %
Day 3	94.39 %
Average	93.02 %

The characteristics of the unsuccessfully classified flows were examined. The unsuccessfully classified flows are

interesting because they somehow differ from the typical behavior of the root class. We back-traced the unsuccessfully classified flow records back to the original data. The results were once again surprising. A clear majority of the unsuccessfully classified flows belonged to the WWW-type traffic profile.

TABLE III. COUNT AND DISTRIBUTION OF UNSUCCESSFULLY CLASSIFED FLOW RECORDS

	Day 1	Day 2	Day 3
Flow count	295884	332649	229839
Original Class WWW	90.86 %	80.16 %	90.33 %
Original Class Other	9.14 %	19.84 %	9.67 %

This might mean that these flows belong to some WWW-based service, which is clearly different from the mass, or even more interestingly, they might be somehow malicious. A clear benefit is also the fact that the amount of data for further analysis is significantly smaller than the amount we started with. It has come down from over 40 million flows to a few hundred thousand rows of interesting data. Clearly, the successfully classified data cannot be totally ignored in the belief that it does not hold in any false positives. However, the first step is to try to identify the true negatives or false negatives from the unsuccessfully classified data. Table 3 illustrates the total number of unsuccessfully classified rows per daily datasets, together with the proportions of the class in the original data.

As a whole the framework performed very well, and therefore can be recommended for feature identification and characteristics examination purposes of IPFIX flow data. The main benefit of the framework in general is that it provides a solid and defined workflow for classification analysis. In many cases the actual workflow is lost behind the results, and the repeatability of the results is therefore compromised. We designed the framework to be solid, yet flexible enough so it wouldn't cause too many restrictions to the analysis work. The framework does not restrict the classification algorithm selection in any way. If some other classifier is used, the top-level framework is applicable as it is. The classification algorithm selection affects to training and classification execution phases. The cornerstones of the framework are the basic profiling phase and the classification training phase. These are clearly also the places for fine-tuning the framework.

VII. CONCLUSION AND THE FUTURE

The goal was to define a framework for classifying IPFIX flow data with KNN Classifier and prove its functionality. The overall classification success ratios were at a very good and promising level throughout the research. Over 90% classification accuracy with a considerably large amount of flow data is an indication of a very good performance. We used cross-validation in the classifier training phase and a three-way classification setup in the actual classification phase in order to prove the classification framework to be solid and robust. In conclusion, we can state that the framework performed well and the results were very promising. They have certainly given us a boost to continue our research.

There are several interesting starting points for the further analysis. Our research is related to information security and the analysis of the WWW-type traffic has many interesting information security aspects, as it is a commonly used and therefore misused service. It utilizes mainly standard port numbering, which means that those ports are typically left open in firewall configurations, thus leaving some space in which the misusers and attackers can operate. In future research we aim to detect the misuse inside WWW-type traffic by trying to point out what is normal and what is not. We aim to do this with total anonymity so that misuse identification and the results analysis is not illegal or harmful to anyone.

Our intention is also to examine the limits of the KNN classifier by further training the classifier. Utilizing the framework with other types of classification methods is also in the scope of interest in the future research. There are public datasets available that can be used as reference datasets and for further validation of the framework.

REFERENCES

[1] IPFIX Working Group, http://datatracker.ietf.org/wg/ipfix/charter/, retrieved: January, 2012

[2] IPFIX Specifications, http://datatracker.ietf.org/wg/ipfix/, retrieved: January, 2012

[3] R. Bendrath. "Global technology trends and national regulation: Explaining Variation in the Governance of Deep Packet Inspection", ISA's 50th Annual Convention "Exploring The Past, Anticipating The Future", New Ypork city, NY, USA, Feb 15, 2009

[4] T. Karagiannis, K. Papagiannaki and M. Faloustos. "BLINC: Multilevel Traffic Classification in the Dark", SIGCOMM'05, Philadelphia, Pennsylvania, USA, August, 2005, pp. 22–26

[5] P. Kumpulainen, K. Hätönen, O. Knuuti and T. Alapaholuoma, "Internet Traffic Clustering Using Packet header Information", Joint International IMEKO Symposium Jena, 2011

[6] A. W. Moore and D. Zuev. "Internet Traffic Classification Using Bayesian Analysis Techniques". SIGMETRICS'05, Banff, Alberta, Canada, June 6-10. 2005

[7] A. A. Yarifard and M. H. Yaghmaee. "The Monitoring System Based on Traffic Classification", World Applied Sciences Journal 5:, 2008, pp. 150-160

[8] Thuy T.T. Nguyen and G. Armitage. "A Survey of Techniques for Internet Traffic Classification using Machine Learning", 2008, ISSN: 1553-877X, pp. 56-76

[9] RFC3954 – Cisco Netflow 9, http://www.ietf.org/rfc/rfc3954.txt, retrieved: January, 2012

[10] MAJI, http://research.wand.net.nz/software/maji.php, retrieved: January, 2012

[11] Libtrace, http://research.wand.net.nz/software/libtrace.php, retrieved: January, 2012

[12] SQLite, http://www.sqlite.org/, retrieved: January, 2012

[13] R. Herbrich. "Learning Kernel Classifiers. Theory And Algorithms". The MIT Press. Cambridge, Massachussets, London, England. ISBN 0-262-08306-X, 2002

[14] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. Mclachlan, A. Ng, B. Liu, P. S. Yu, Z-H. Zhou, M. Steinbach, D. J. Hand and D Steingerg. "Top 10 Algorithms in Data Mining", Survey Paper, DOI 10.1007/s10115-007-0114-2, pp. 14:1–37

 ISBN: 978-1-61208-186-1