A method for classification of network traffic based on C5.0 Machine Learning Algorithm

Tomasz Bujlow, Tahir Riaz, Jens Myrup Pedersen
Section for Networking and Security, Department of Electronic Systems
Aalborg University, DK-9220 Aalborg East, Denmark
{*tbu, tahir, jens*}*@es.aau.dk*

***Abstract*—Monitoring of the network performance in a high-speed Internet infrastructure is a challenging task, as the requirements for the given quality level are service-dependent. Therefore, the backbone QoS monitoring and analysis in Multi-hop Networks requires the knowledge about the types of applications forming the current network traffic. To overcome the drawbacks of existing methods for traffic classification, usage of C5.0 Machine Learning Algorithm (MLA) was proposed. On the basis of the statistical traffic information received from volunteers and C5.0 algorithm, we constructed a boosted classifier, which was shown to have the ability to distinguish between 7 different applications in the test set of 76,632–1,622,710 unknown cases with average accuracy of 99.3–99.9 %. This high accuracy was achieved by using high quality training data collected by our system, a unique set of parameters used for both training and classification, an algorithm for recognizing flow direction and the C5.0 itself. The classified applications include Skype, FTP, torrent, web browser traffic, web radio, interactive gaming and SSH. We performed subsequent tries using different sets of parameters and both training and classification options. This paper shows how we collected accurate traffic data, presents arguments used in classification process, introduces the C5.0 classifier and its options, and finally, evaluates and compares the obtained results.**

***Index Terms*—traffic classification, computer networks, C5.0, Machine Learning Algorithms (MLAs), performance monitoring**

I. Introduction

One of the most important challenges in the network monitoring is how to measure the performance of high-speed Multi-hop Networks in a centralized manner. Each network carries data for numerous different kinds of applications, which have different performance requirements. Therefore, providing the information about the quality level requires the knowledge about what kinds of data are flowing in the network at the present time. Most of the current methods for traffic classification use the concept of a flow defined as a group of packets having the same end IP addresses, using the same transport protocol, and its port numbers. Flows are considered as bidirectional – packets going from the local machine to the remote server and from the remote server to the local machine are a part of the same flow.

Using application ports for traffic classification is a very simple idea widely used by network administrators to limit the traffic generated by worms and unwanted services. This method is very fast and can be applied to almost all the routers and layer-3 switches existing on the market. Apart from its universality, this method is very efficient to classify some protocols operating on fixed port numbers. Using it, however, gives very bad results in detection of protocols using dynamic port numbers, like P2P and Skype [1]–[3]. The second drawback is not less severe: many scam applications use well-known port numbers to be treated in the network with a priority. Deep Packet Inspection (DPI) solutions are quite slow and require a lot of processing power [1], [3]. Furthermore, they rely on inspecting the user data and, therefore, privacy and confidentiality issues can appear [1]. Application signatures for every application must be created outside the system and kept up to date [1], which can be problematic. Worse, encryption techniques make DPI in many cases impossible.

Machine Learning Algorithms like K-Means, Naive Bayes Filter, C4.5, J48, Random Forests have much wider coverage. They can be used in any point of the network, providing very fast statistical detection of the application, to which the traffic belongs. Achievable detection rate correctness is over 95 % [1], [2], [4]–[9]. All the MLAs require significant amounts of training data for initial learning. The precision of the future classification by MLAs depends heavily on the quality of the training data. This paper introduces the usage of C5.0 in traffic classification and shows that this C4.5 successor is able to offer the classification accuracy of above 99 %.

The remainder of this document describes related previous work, gives an overview of our system, our method for collecting precise training data and isolating set of arguments used for classification, and then focuses on C5.0. The accuracy of the classification by C5.0 and the speed of generating the classifier was assessed when using various set of classification arguments and program options. Subsequently, the obtained results were presented and discussed.

II. Related Work

It was demonstrated in [1] that all the P2P applications behave similarly, so it is possible to use statistical analysis to detect even unknown applications. Several tries were made to classify accurately P2P and Skype traffic using older implementations of MLAs, like REPTree, C4.5, or J48. In [1], the authors proposed a few simple algorithms based on REPTree and C4.5, which are being able to classify P2P traffic using the first 5 packets of a flow. Their method based on C4.5 performed highly accurately (97 % of P2P traffic was classified

properly), but the accuracy was not tested when starting packets from the flow were lost. Furthermore, the attribute set used for classification contained source and destination port numbers, what could make the classifier closely related to the current assignment of port numbers to particular applications in the training data.

Another approach to classify P2P applications was taken in [3] using a Java implementation of C4.5 called J48 to distinguish between 5 different applications. The authors tried to skip a number of packets in the beginning of the flow ranging from 10 to 1000 and they obtained only a little fluctuation in performance, with classification accuracy over 96 %. It was shown in [10] that the original C4.5 and J48 perform much different on relatively small or noisy data sets (the accuracy of J48 and C5.0 was in tested cases similar, and worse than C4.5). J48 processing using statistics based on sizes was implemented in [11] for detection of BitTorrent and FTP traffic, reaching the accuracy of around 98 %. This publication showed that behavior of data parameters contained in encrypted and unencrypted traffic generated by the same application looks almost the same. Moreover, it was shown that zero-payload packets (ACK) can distort statistics based on sizes.

In [12], many different mechanisms of classification of the network traffic were evaluated, including C5.0. The achieved accuracy was around 88–97 % on traffic belonging to 14 different application classes. This not very high classification accuracy was probably partly due to preparing both training and test cases, where the decision attribute (application name) was obtained by DPIs (PACE, OpenDPI and L7-filter). These DPI solutions use multiple algorithms (including statistical analysis) to obtain the application name. Therefore, both training and test data were in some degrees inaccurate, which caused also more errors from the side of C5.0.

III. Overview of the Methods

In our research, the C5.0 classifier was intended to be a part of a system for Quality of Service (QoS) measurements in the core of the network [13]. The first task is to recruit volunteers from the users in the network, in which the system will be installed. The volunteers install on their computers a client program, which captures the relevant traffic information and submits the data to the server. On the server, these data are used to generate per-application traffic statistics. C5.0 Machine Learning Algorithm uses these statistics to learn how to distinguish between different types of applications and generate classification rules (decision trees). In our research, we focused on 7 different groups of applications instead of individual applications, because the QoS requirements within each group are similar (like for Firefox, Opera or Google Chrome web browsers).

The challenging task is to inspect nearly in real-time significant amount of traffic in the core of high-speed networks. Such systems deal with huge amounts of data and, therefore, only selected flows can be inspected due to memory and processing power limitations for quality assessment. Inspecting one or few flows per user at a time is enough, since when a user experiences problems, they usually concern all user's network activity. For better adjustment to applications used in different networks, the classifier was designed to be network-dependent, so it should be trained in each network independently. When the relevant flows are captured, per-flow statistics need to be generated. There are two kind of statistics generated at this step: used to determine the kind of application associated to that flow, and used to assess the QoS level in a passive way. The system uses the classification rules previously generated by C5.0 together with the first type of statistics to find out to which application the flow belongs. Then, on the basis of the kind of the application, the system determines acceptable ranges of values of the relevant QoS parameters. The last step is to check if the current values (obtained from flow statistics or in an active way) match the expected ones. If not, the quality of the given service is considered as degraded.

The subsequent paragraphs contain detailed description of our methods regarding:

- Collecting accurate training and test data by our Volunteer-Based System
- Criteria for the data flows used in our experiment
- Processing the flows and extracting the statistics
- Defining sets of classification arguments
- Assessing the accuracy of C5.0 while using various classification options

IV. Obtaining the Data

A good solution for obtaining accurate training data can rely on collecting the flows at the user side along with the name of the associated application. We did this using our Volunteer-Based System. The basic idea and the design of the system were described in [14] and our current implementation in [15]. The system consists of clients installed on users' computers, and a server responsible for storing the collected data. The task of the client is to register the information about each flow passing the Network Interface Card (NIC), with the exception of traffic to and from the local subnet, to prevent capturing transfers between local peers. The following flow attributes were captured: start and end time of the flow, number of packets, local and remote IP addresses, local and remote ports, transport protocol, name of the application and client, which the flow belongs to. Apart from the information on the flow itself, the client also collected information about all the packets associated with each flow. These packet parameters were: direction, size, TCP flags, and relative timestamp to the previous packet in the flow. Information was then transmitted to the server, which stored all the data for further analysis in a MySQL database.

Another small software was developed for generating training and test files for the C5.0 classifier from the collected data. The following application groups were isolated: Skype main voice flow, FTP transfers (both uploads and downloads), torrent transfers, web browser traffic (except web radio), web radio traffic, the interactive game America's Army and SSH

Table I
REQUIREMENTS FOR DIFFERENT FLOWS

Group	Requirement
Skype	protocol name = 'UDP' AND application name = 'Skype' AND no. of packets $\geq$ 200
FTP	application name = 'Filezilla'
Torrent	application name = 'uTorrent'
Web	(application name = 'firefox' OR application name = 'chrome' OR application name = 'opera') AND remote IP $\neq$ '195.184.101.203'
Web radio	client id = 3 AND application name = 'chrome' AND remote IP = '195.184.101.203'
Game	application name = 'AA3Game' AND protocol name = 'UDP'
SSH	application name = 'Putty'

traffic. The requirements needed to be fulfilled by the traffic flows associated with each group are specified in Table I.

Because of dynamic switching between the flows, the method had to be able to inspect a flow starting from any time point. For performance reasons, it is not possible to store all the flows in the database and to start inspection of the chosen one from its beginning. So, it had to be possible to assess the flow on the basis of the given number of packets or seconds from the middle of that flow. We assumed that the flow characteristics based on packet sizes within a network are independent of current conditions, contrary to the flow characteristics based on time parameters (which change quickly during e.g. congestion). Therefore, our method used the concept of a probe equals to a particular number of captured packets instead of particular number of seconds. The probability of catching initial packets of each flow is very low due to dynamic switching between the flows. Moreover, count and size characteristics are different for the initial and for the remaining packets in the flow. To not disturb the accuracy of the classifier by statistics obtained from initial packets, we decided to ignore in the experiment ten initial packets of each flow, even if they were captured and stored. This feature excluded from the experiment flows possessing less than 15 packets, but this limitation was reasonable, because the QoS performance measurements rely in our case on long flows.

The direction of the flow was recognized on the basis of proportions of inbound to outbound payload bytes of the classified flow – the higher value was always considered as belonging to the inbound traffic. This way, streams with asymmetric load were always classified in the same way and we avoided noise affecting the accuracy.

We needed to find out what number of packets from the flow is needed to perform accurate classification. Each flow was divided into X groups of Y packets, where Y depends on the current iteration (we tested our algorithm on groups of 5, 10, 15,..., 90 packets), and X is a count of obtained groups. The dependency between X and Y is evident: more packets in a group means less groups in total (and, therefore, less training cases), but higher accuracy of statistics creating each particular case. Obtained groups were divided into 2 disjoint sets used later to generate statistics.

V. CLASSIFICATION ATTRIBUTES

Each group from these 2 disjoint sets was used to generate one (respectively training and testing) case for the classifier. This way we never used the same cases for both training and classification. The attributes were divided into 2 sets. Set A contained 32 general continuous attributes based only on packet count and sizes, plus the target attribute. All the size parameters were based on the real payload length, not the packet length. To improve the ability to classify the encrypted traffic, the outer TCP/IP (40 B) or UDP/IP (28 B) header was removed, leaving only the data part. This set of parameters consists of:

- Number of inbound / outbound / total payload bytes in the sample
- Proportion of inbound to outbound data packets / payload bytes
- Mean, minimum, maximum first quartile, median, third quartile, standard deviation of inbound / outbound / total payload size in the probe
- Ratio of small inbound data packets containing 50 B payload or less to all inbound data packets
- Ratio of small outbound data packets containing 50 B payload or less to all outbound data packets
- Ratio of all small data packets containing 50 B payload or less to all data packets
- Ratio of large inbound data packets containing 1300 B payload or more to all inbound data packets
- Ratio of large outbound data packets containing 1300 B payload or more to all outbound data packets
- Ratio of all large data packets containing 1300 B payload or more to all data packets
- Application: skype, ftp, torrent, web, web_radio, game, ssh

Set B contained 10 protocol-dependent attributes:

- Transport protocol: TCP, UDP
- Local port: well-known, dynamic
- Remote port: well-known, dynamic
- Number of ACK / PSH flags for the inbound / outbound direction: continuous
- Proportion of inbound packets without payload to inbound packets: continuous
- Proportion of outbound packets without payload to outbound packets: continuous
- Proportion of packets without payload to all the packets: continuous

By dividing the classification attributes, the research demonstrated how much the accuracy of the classifier depends on including set B into the classification. It is worth mentioning that the attributes contained in set B are very general: no real port numbers were stored, but only the information if the number is below 1024 (well-known) or above (dynamic). This way, at first, we did not influence the speed of the classifier by dividing the cases for each local and remote port number (resulting in generating port-based classifier), but still we were able to provide general

Table II
PERCENT OF CLASSIFICATION ERRORS WHEN USING DIFFERENT C5.0 OPTIONS

No. of pckts / case	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
No. of cases	1622710	791552	520661	384757	304388	251420	213810	185686	163715	146248	131818	119710	109449	100908	93572	87225	81546	76632
Decision trees [A]	5.0	4.0	3.5	3.2	3.2	3.0	3.1	2.9	2.9	3.0	2.9	3.0	3.2	3.0	3.0	3.0	3.1	3.0
Decision trees [A+B]	0.5	0.4	0.7	0.5	0.5	1.0	0.5	0.9	0.3	0.4	0.5	0.4	0.3	0.4	0.4	0.3	0.3	0.3
Rules [A]	5.5	5.4	3.7	3.1	3.2	4.1	3.4	3.0	3.9	3.2	2.9	3.1	3.0	2.9	2.9	2.9	2.8	2.9
Rules [A+B]	0.5	0.4	0.6	0.4	0.4	1.0	0.3	0.3	0.3	0.4	0.2	0.4	0.3	0.2	0.2	0.3	0.2	0.3
Boost [A]	5.0	3.9	3.5	3.0	2.9	2.6	2.7	2.8	2.8	2.9	4.2	4.5	3.1	4.0	4.4	4.3	3.8	2.9
Boost [A+B]	0.4	0.4	0.3	0.4	0.4	0.3	0.2	0.2	0.2	0.7	0.2	0.2	0.2	0.2	0.1	0.2	0.3	0.2
Softening [A]	4.9	4.0	3.5	3.2	3.3	3.0	3.2	2.8	3.1	3.0	3.0	3.0	3.1	3.0	3.0	2.9	3.0	3.0
Softening [A+B]	0.5	0.4	0.7	0.5	0.5	0.9	0.5	0.9	0.3	0.4	0.5	0.4	0.3	0.4	0.4	0.3	0.3	0.3

information about the application model: client-server or peer-to-peer. Secondly, services can use different port numbers, but still they should be classified correctly, as the servers in the client-server model usually use only well-known port numbers, and the clients use dynamic ones. By avoiding using the port numbers, our solution should be able to identify accurately also encrypted traffic. Zero-payload packets (ACK) were treated in a special way by assignment to their own classification argument.

VI. C5.0-BASED CLASSIFIER

The C5.0 algorithm is a new generation of Machine Learning Algorithms (MLAs) based on decision trees [16]. It means that the decision trees are built from the list of possible attributes and the set of training cases, and then the trees can be used to classify the subsequent sets of test cases. C5.0 was developed as an improved version of a well-known and widely used C4.5 classifier and it has several important advantages over its ancestor [17]. The generated rules are more accurate and the time used to generate them is lower (even around 360 times on some data sets). In C5.0, several new techniques were introduced:

- Boosting: several decision trees are generated and combined to improve the predictions
- Variable misclassification costs: that makes possible to avoid errors, which can result in a harm
- New attributes: dates, times, timestamps, ordered discrete attributes
- Values can be marked as missing or not applicable for particular cases
- Supports sampling and cross-validation

The C5.0 classifier contains a simple command-line interface, which was used by us to generate the decision trees, rules and finally test the classifier. In addition, free C source code for including C5.0 classifier in external applications is available on the website of C5.0. Detailed description of C5.0 and all its options and abilities is published in the tutorial [18].

VII. RESULTS

The training cases were provided to the C5.0 classifier to generate decision trees or classification rules. Then, the decision trees or the rules were used to classify the test cases. The experiment was repeated multiple times, each time using different sets of training and test cases (dependent on the number of packets used to create the case), different set of attributes used for classification (set A, or set A plus B), and different classification options (normal, rules generating, boosting, softening thresholds). We tested both the error rates of the provided classifiers (Table II) and the time needed to construct them (Figure 2). The average error rates of the classifiers are shown in Figure 1 and the misclassification table for the classifier with the lower error rate (boosted classifier using both A and B sets of classification attributes, 75 packets used to construct each case) is presented in Figure 3. The experiment resulted in several important conclusions. First of all, using extended set of classification attributes (A + B) containing protocol-dependent attributes, we achieved lower bottom error rate (0.1 %) than using only size-based attributes from set A (2.7 %).

The time used to construct classifiers from the extended set of attributes was also lower than when using only set A. Both these observations were completely independent of classification options. The lowest error rate of 0.1 % was achieved by using the boosted classifier, comparing to 0.3 % error rate when using the standard classification without any options. However, creating the boosted classifier took around 10 times more time than creating the standard classifier. Furthermore, our research demonstrated that creating the rules instead of decision trees, or using softened thresholds had no or only a little impact on the error rate, while it extended dramatically the time used to construct the classifier.

We also measured which way of training the classifier is the most optimal. The research showed that the classification error rate was the highest when using numerous cases, each created using statistics derived from 5 subsequent packets. The low precise statistics constructed from small number of packets were not sufficient to make the classifier accurate, even, if the count of them was significant. The classification error was decreasing while we were increasing the number of packets from which the statistics were generated, and it stabilized when we used 35 or more packets. Further increasing of the number of packets used to construct the case further did not

improve the accuracy significantly, probably because it was compensated by a smaller number of provided training cases.

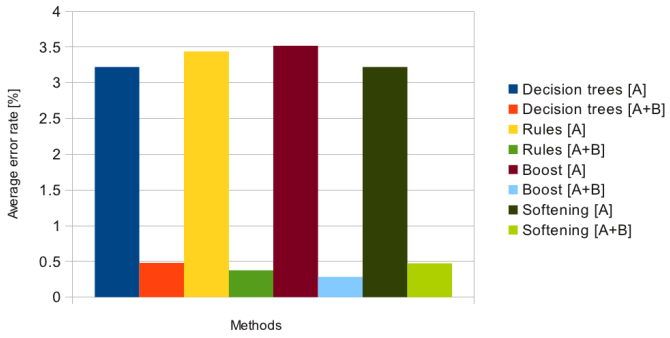

Figure 1. Average error rates of the classifiers

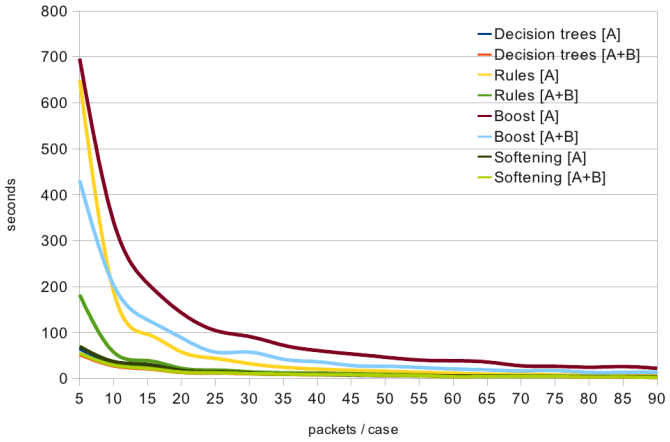

Figure 2. Time spent for generating the classifier

(a)	(b)	(c)	(d)	(e)	(f)	(g)	<-classified as
650					3		(a): class skype
	196	68					(b): class ftp
	19	83929	3				(c): class torrent
			6799				(d): class web
			8	388			(e): class web_radio
16		9			1419		(f): class game
			7			58	(g): class ssh

Figure 3. Misclassification table, the best case

VIII. Conclusion

This paper presents a novel method based on C5.0 MLA for distinguishing different kinds of traffic in computer networks. It was demonstrated that our method is feasible to classify the traffic belonging to 7 different applications with average accuracy of 99.3–99.9 %, when using accurate data sets for both training and testing the boosted classifier. Our results proved that the classifier is able to distinguish traffic which appears to be similar, like web browser traffic and a radio streamed via a web page. The classifier did not have problems with distinguishing interactive traffic: Skype, game and SSH. We observed, however, that FTP and Torrent file transfers have very similar flow characteristics and, therefore, a significant number of packets were misclassified between these two classes. Our method is a field for more experiments and further improvements. In this research both the training and test data sets were disjoint, but collected from the same users. As the next step, we consider to involve numerous users to assess the accuracy using data sets obtained from different networks.

References

[1] Jun Li, Shunyi Zhang, Yanqing Lu, and Junrong Yan. Real-time P2P traffic identification. In *Proceedings of the IEEE Global Telecommunications Conference (IEEE GLOBECOM 2008)*, pages 1–5. IEEE, New Orleans, Louisiana, USA, December 2008. DOI: 10.1109/GLOCOM.2008.ECP.475.

[2] Jing Cai, Zhibin Zhang, and Xinbo Song. An analysis of UDP traffic classification. In *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pages 116–119. IEEE, Nanjing, China, November 2010. DOI: 10.1109/ICCT.2010.5689203.

[3] Ying Zhang, Hongbo Wang, and Shiduan Cheng. A Method for Real-Time Peer-to-Peer Traffic Classification Based on C4.5. In *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pages 1192–1195. IEEE, Nanjing, China, November 2010. DOI: 10.1109/ICCT.2010.5689126.

[4] Riyad Alshammari and A. Nur Zincir-Heywood. Machine Learning based encrypted traffic classification: identifying SSH and Skype. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009)*, pages 1–8. IEEE, Ottawa, Ontario, Canada, July 2009. DOI: 10.1109/CISDA.2009.5356534.

[5] Sven Ubik and Petr Žejdl. Evaluating application-layer classification using a Machine Learning technique over different high speed networks. In *Proceedings of the Fifth International Conference on Systems and Networks Communications (ICSNC)*, pages 387–391. IEEE, Nice, France, August 2010. DOI: 10.1109/ICSNC.2010.66.

[6] Riyad Alshammari and A. Nur Zincir-Heywood. Unveiling Skype encrypted tunnels using GP. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, Barcelona, Spain, July 2010. DOI: 10.1109/CEC.2010.5586288.

[7] Li Jun, Zhang Shunyi, Lu Yanqing, and Zhang Zailong. Internet Traffic Classification Using Machine Learning. In *Proceedings of the Second International Conference on Communications and Networking in China (CHINACOM '07)*, pages 239–243. IEEE, Shanghai, China, August 2007. DOI: 10.1109/CHINACOM.2007.4469372.

[8] Yongli Ma, Zongjue Qian, Guochu Shou, and Yihong Hu. Study of Information Network Traffic Identification Based on C4.5 Algorithm. In *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pages 1–5. IEEE, Dalian, China, October 2008. DOI: 10.1109/WiCom.2008.2678.

[9] Wei Li and Andrew W. Moore. A Machine Learning Approach for Efficient Traffic Classification. In *Proceedings of the Fifteenth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'07)*, pages 310–317. IEEE, Istanbul, Turkey, October 2007. DOI: 10.1109/MASCOTS.2007.2.

[10] Samuel A. Moore, Daniel M. D'addario, James Kurinskas, and Gary M. Weiss. Are Decision Trees Always Greener on the Open (Source) Side of the Fence? In *Proceedings of the 5th International Conference on Data Mining (DMIN'2009)*, pages 185–188. The Lancaster Centre for Forecasting, Las Vegas, Nevada, USA, July 2009.

[11] Jason But, Philip Branch, and Tung Le. Rapid identification of BitTorrent Traffic. In *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 536–543. IEEE, Denver, Colorado, USA, October 2010. DOI: 10.1109/LCN.2010.5735770.

[12] Daniele De Sensi, Marco Danelutto, and Luca Deri. Dpi over commodity hardware: implementation of a scalable framework using fastflow. Master's thesis, Università di Pisa, Italy, 2012. Accessible: http://etd.adm.unipi.it/t/etd-02042013-101033/.

[13] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. A method for Assessing Quality of Service in Broadband Networks. In *Proceedings of the 14th International Conference on Advanced Communication Technology (ICACT)*, pages 826–831. IEEE, Phoenix Park, PyeongChang, Korea, February 2012. Accessible: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6174795.

[14] Kartheepan Balachandran, Jacob Honoré Broberg, Kasper Revsbech, and Jens Myrup Pedersen. Volunteer-Based Distributed Traffic Data Collection System. In *Proceedings of the 12th International Conference on Advanced Communication Technology (ICACT 2010)*, volume 2, pages 1147–1152. IEEE, Phoenix Park, PyeongChang, Korea, February 2010.

[15] Tomasz Bujlow, Kartheepan Balachandran, Tahir Riaz, and Jens Myrup Pedersen. Volunteer-Based System for classification of traffic in computer networks. In *Proceedings of the 19th Telecommunications Forum TELFOR 2011*, pages 210–213. IEEE, Belgrade, Serbia, November 2011. DOI: 10.1109/TELFOR.2011.6143528.

[16] Information on See5/C5.0 – RuleQuest Research Data Mining Tools, 2011. [Online]. Available: http://www.rulequest.com/see5-info.html.

[17] Is See5/C5.0 Better Than C4.5, 2009. [Online]. Available: http://www.rulequest.com/see5-comparison.html.

[18] C5.0: An Informal Tutorial, 2011. [Online]. Available: http://www.rulequest.com/see5-unix.html.