Impact of Asymmetry of Internet Traffic for Heuristic Based Classification

Chris Richter richter@imn.htwk-leipzig.de
Faculty of Computer Science
HTWK Leipzig
Leipzig, 04277, Germany

Michael Finsterbusch finster@imn.htwk-leipzig.de
Faculty of Computer Science
HTWK Leipzig
Leipzig, 04277, Germany

Klaus Hänßgen haenssge@imn.htwk-leipzig.de
Faculty of Computer Science
HTWK Leipzig
Leipzig, 04277, Germany

Jean-Alexander Müller jeanm@hft-leipzig.de
Department of Communication and Computer Science
Hochschule für Telekommunikation Leipzig
Leipzig, 04277, Germany

Abstract

Accurate traffic classification is necessary for many administrative networking tasks like security monitoring, providing Quality of Service and network design or planning. In this paper we illustrate the accuracy of 18 different machine learning algorithms with different statistical parameter combinations. Additionally, we divide the statistical parameters into upstream and downstream to observe the influence of the protocol inherent differences of client and server behaviour for traffic classification. Our results show that this differentiation can increase the protocol detection rate and decrement the processing time.

Keywords: flow classification, Internet traffic, traffic identification.

1. INTRODUCTION

For operation, management and design of communication networks, advanced knowledge of transmitted protocols and applications as well as their behaviour are necessary. This detailed information can be provided by traffic classification. Network traffic classification, or classification of applications, is the process of identifying the type of protocols or applications which generate particular network flows. Scope of applications for traffic classification are, for example, the providing of information about future traffic evolution (trend analysis), traffic engineering, intrusion detection and prevention, content control/filtering, monitoring and lawful interception.

In general, there are four kinds of traffic classification methods. The oldest and most commonly used method is the *port based* approach. This uses the well-known port numbers of the TCP/UDP protocols assigned by the IANA. Another method used is *protocol decoding*. It is based on stateful reconstruction of sessions and application information from packet content. It identifies protocols by their characteristic protocol headers (magic numbers, incrementing counters, session identifiers, etc.), packet sequences, etc. so it avoids needing to trust in port numbers. This method is often used only for dedicated popular protocols, e.g., HTTP and mail protocols like in Cisco's Network Based Application Recognition (NBAR) [1]. The third method is the *pattern* or *signature based* approach [2]. This method uses application specific signatures and searches for those in the protocol header and content to identify the application.

The fourth method is based on the *machine learning* approach. This method uses machine learning algorithms as used in data mining to identify applications by characteristic packet or flow statistics. The advantage of this approach is that the algorithms can be trained with real network traffic. If a protocol changed or a new protocol appeared, it is easy to repeat the training to update the protocol identifier. It can possibly be used to identify some encrypted protocols. A problem of this method is to find the proper parameters and effective machine learning algorithms.

In this paper we evaluate the impact of the network protocols asymmetry behaviour of Internet traffic for classification with a large set of machine learning algorithms and parameters. In our investigations we define asymmetry behaviour as the different behaviour of the protocols in their downstream and upstream directions. The machine learning approach has been discussed in numerous papers [3, 4, 5, 6, 7, 8], but with focus on just one algorithm. Furthermore, these approaches are mostly used for non real-time or offline network traffic analysis [3, 4, 6]. Besides the evaluation of how parameter reduction can influence the accuracy of classification, we aim to evaluate the influence of the classification runtime. These results may reveal an opportunity for using machine learning algorithms in future real-time classification.

The remainder of this paper is structured as follows: Section 2 contains a description of the experimental setup of our research, and Section 3 focus on the results of the traffic classification. In the following Section 4, we are describing the influence of parameter reduction according to the classification accuracy and the time consumption. Section 5 compares our results with the results of other studies. Finally, Section 6 provides a conclusion and the direction for future work.

2. EXPERIMENTAL SETUP

We used different network traffic traces in PCAP (packet capture library) format [9] to test and train the investigated machine learning algorithms of this study. To extract the necessary detailed protocol information and the required statistical parameters of these traces, we used our own developed tool described in [10], because available tools like GTVS (Ground Truth Verification System) [11] do not fulfil our constraints for the automatic traffic labelling. To describe the protocol characteristics, we used 40 different parameters, which are a subset of the six parameter classes: packet count, interarrival time, payload size, flow duration, bulk mode and idle mode. Some of these parameters characterize the whole flow, while the remaining parameters characterize the flow separately for upstream and downstream. This separation is done to observe the impact of the asymmetric behaviour of the network protocols for the classification. More detailed description of the 40 parameters can be found in [10].

The network traffic classification based on the protocol characteristics is done with 18 different machine learning algorithms – also called classifiers. These classifiers are provided by the WEKA software suite [12] and we treat them as black-box classifiers. The selection of these 18 classifiers is described in [10]. For the automated supervised training (Phase 1) as well as for testing (Phase 2) and protocol classification, respectively, we build a test-suite on top of WEKA.
The training data contains all statistical parameters and the associated protocol. Therefore, we can use the supervised learning approach for the machine learning algorithms. The testing data contain only the statistical parameters. During the training the classifiers generate a classifier model; this model can be used afterwards for testing in Phase 2 with different traffic. The classification accuracy of the classifiers can be validated by comparing the prediction of the classifier with the known protocol information. Due to a lack of publicly available network traces with full payload, which is necessary to evaluate the exact protocol or application of the traffic, we generated different kinds of traffic. More information to the used traffic can be found in [10].

The process of generating the classifier models is deterministic for our training data and classifiers, with the exception of the AttributeSelectedClassifier. Thus, the particular generated classifier models are always the same for a given training set. The test results computed by the classifiers with their applied model are deterministic, too.

3. RESULTS

Table 1 contains the results of this study the filled symbols are the added results from Section 4. It shows if it is possible for an algorithm to detect a protocol with an accuracy greater than or equal to 90% with our selected parameters. Furthermore, Table 1 differentiates the results into three categories to observe the flow-direction asymmetry of the investigated network protocols:

- *full*: all 40 parameters
- *splitting*: parameters which are computed separately for the directions upstream and downstream (26 parameters)
- *no splitting*: parameters which are calculated for the whole flow (14 parameters)

WEKA Classifier Name	Bittor-rent	eDon-key	Flash	HTTP	IMAP	Oscar	POP3	RTP	SIP	SMTP
AttributeSelectedClassifier	○ Δ □	▲	○ Δ □			○ Δ □	○ Δ ■	○ Δ ■	○ Δ □	● ■
Bagging	● Δ □	▲ ■	○ Δ □	▲		○ Δ □	○ Δ □	○ Δ ■	▲ □	○ Δ ■
BayesNet	○ Δ □		○ Δ □	▲		○ Δ □	○ Δ ■	● ▲ □	○ Δ □	○ Δ □
DataNearBalancedND	○ Δ □	▲ ■	● Δ □	● ▲ □	●	○ Δ □	○ ▲ ■	○ ▲ ■	○ Δ □	● Δ □
DecisionTable	○ Δ ■		○ Δ □			▲	● ■	○ Δ ■	● ■	
FilteredClassifier	○ Δ ■	● ▲ ■	○ Δ ■	▲ □		○ Δ □		● ▲ ■	○ Δ ■	
J48	● Δ □	● ▲	● Δ □	● ▲ ■		○ Δ □	○ Δ □	○ ▲ □	○ Δ □	○ ▲ ■
J48graft	○ Δ □	● ▲ ■	○ Δ □	● ▲ ■		● Δ □	○ Δ ■	○ Δ □	○ Δ □	● ▲ ■
NaiveBayes	○ Δ □		○ Δ ■	● ■	■		● ▲ ■	● ▲ □	○ ▲ ■	●
NaiveBayesUpdateable	○ Δ □		○ Δ ■	● ■	■		● ▲ ■	● ▲ □	○ ▲ ■	●
nestedDichotomies.ND	○ ▲ ■	▲ ■	● ▲ □	● ▲ ■		● Δ ■	○ Δ ■	○ ▲ □	● ▲ ■	● Δ ■
OneR	○ Δ ■	● ▲	○ Δ □					● ▲ □	○ Δ ■	
PART	○ Δ □	▲	○ Δ □	● ▲ □		○ Δ □	○ Δ □	○ Δ □	○ Δ □	○ Δ ■
RandomCommittee	○ Δ □	● ▲ ■	○ Δ □	○ Δ □		○ Δ □	○ Δ □	○ Δ □	○ Δ □	○ Δ □
RandomForest	○ Δ □	▲ ■	○ Δ □	○ Δ □		○ Δ □	○ Δ □	○ Δ □	○ Δ □	○ Δ □
RandomSubSpace	○ Δ □	● ▲ ■	○ Δ □	○ Δ □		○ Δ □	○ Δ □	○ Δ □	○ Δ □	○ Δ □
RandomTree	○ Δ □	▲ ■	○ Δ □	● Δ □	●	● ▲ □	○ ▲ ■	○ Δ □	○ ▲ □	○ ▲ □
REPTree	● Δ □	▲	○ Δ □	▲ □		○ Δ □	○ Δ □	○ Δ ■	▲ □	○ Δ ■

TABLE 1: Protocol classification with greater than or equal to 90% accuracy (○ full, Δ no splitting, □ splitting upstream/downstream; ●▲■ additional from parameter reduction).

3.1 Classification Accuracy

It can be seen from Table 1 that not all algorithms used are suitable for protocol classification with our selected statistical parameters. The classifiers DecisionTable, nestedDichotomies.ND, OneR, NaiveBayes and NaiveBayesUpdateable have low classification accuracy over all protocols. In contrast, the classifiers RandomCommittee, RandomForest and RandomSubSpace have a high classification accuracy on all protocols, except the two protocols eDonkey and IMAP, which were classified by all algorithms with low accuracy.

Also, the results in Table 1 show that the observed protocols have different characteristics, so that some could be detected with high accuracy (Bittorrent, Flash) while others (eDonkey, IMAP) are hard to detect. Because of the specific characteristics, there are also differences in the classification accuracy for the three parameter categories. For example, the HTTP protocol has the best classification results when using the parameters which are computed for the flow ("splitting") with differentiation between upstream and downstream. In contrast, for the protocol POP3 the classification results have a higher accuracy with the combination of all parameters ("full").

3.2 Training and Testing Effort

Table 2 contains the time needed for training and testing. These times are measured by using the HPROF [13] tool for heap and CPU profiling. We measured the CPU usage time for every algorithm and applied the fastest algorithm (REPTree with parameter category "no splitting") as reference to scale the timing results.

As we can see in Table 2, the amount of time spent for training is much higher than for testing, but this is not a problem in general. Training is done only once, while testing is done on an ongoing basis for protocol classification. All algorithms have very similar CPU time consumption, but four algorithms (BayesNet, NaiveBayes, NaiveBayesUpdateble and ND) have a significantly higher CPU time consumption. This could make these four algorithms unusable for real-time traffic classification.

WEKA Classifier Name	**full**		**splitting**		**no splitting**	
	Train	***Test***	***Train***	***Test***	***Train***	***Test***
AttributeSelectedClassifier	48.3	3.6	32.0	3.3	20.7	3.4
Bagging	180.1	2.1	112.5	2.0	70.6	1.9
BayesNet	62.1	74.9	37.3	46.4	21.4	25.7
DataNearBalancedND	144.2	7.7	74.2	7.4	43.3	6.5
DecisionTable	631.5	2.6	386.3	2.1	199.0	1.7
FilteredClassifier	32.2	3.3	20.7	2.5	12.5	2.3
J48	59.9	3.1	38.4	2.9	25.8	2.9
J48graft	85.8	4.0	54.7	4.2	38.1	3.2
NaiveBayes	86.9	200.7	57.1	133.3	31.3	73.7
NaiveBayesUpdateable	86.7	200.3	57.8	133.5	31.7	73.9
nestedDichotomies.ND	198.5	66.8	139.3	66.6	87.4	64.0
OneR	7.2	1.6	4.8	1.3	3.2	1.1
PART	111.5	2.9	61.9	2.6	51.3	2.5
RandomCommittee	53.3	3.5	40.9	3.0	29.8	.2.6
RandomForest	47.8	3.6	38.1	3.1	31.3	2.6
RandomSubSpace	100.9	2.9	65.7	2.4	38.6	2.3
RandomTree	6.9	1.7	4.8	1.5	4.1	1.3
REPTree	19.6	1.7	11.9	1.4	8.1	1.0

TABLE 2: Classifier time factors.

In Table 2, the time factor for training as well as for testing indicates a relation between the number of parameters and the time consumption of the classifiers. Thus, the time consumption is reduced by using fewer parameters for the protocol classification. Because of these results, we supposed a linear relation between the number of parameters and the time consumption of the classifiers. To verify this supposition, the following Section 4 includes further tests with reduced parameters for the protocol classification.

4. PARAMETER REDUCTION

Because of the test results of the previous Section 3, we decided to evaluate the influence of parameter reduction according to the time consumption of the classifiers and the accuracy of the protocol classification. For the parameter reduction, we divided the parameters of each parameter category ("full", "splitting" and "no splitting") into six different parameter classes. Furthermore, we tested each classifier with all possible 63 combinations of these parameter classes – the 64th combination (all parameter classes removed) was omitted.

4.1 Classification Accuracy

The filled symbols in Table 1 represent the additional classification results of the parameter reduction. In contrast to the results of Section 3, the classification accuracy in the results were increased for all protocols and some classifiers. The improvement of eDonkey and HTTP is significant when comparing all protocols. The classification results of eDonkey show differences in the classification accuracy for the three parameter categories. The best classification results were reached with the parameters which are computed for the whole flow ("no splitting"). In addition, the classification accuracy of the classifier NaiveBayes, NaiveBayesUpdateable and nestedDichotomies.ND could be proliferated. However, Table 1 does not show which parameter combinations were suitable for the best classification results.

Despised the improvement of the classification accuracy by the parameter reduction, the classification accuracy of IMAP is still low. Only the four classifiers DataNearBalancedND, NaiveBayes, NaiveBayesUpdateable and RandomTree are able to classify this protocol with an accuracy of greater than or equal to 90%. We can see the same results on the two classifiers DecisionTable and OneR, having still the lowest classification accuracies according to all ten protocols. Even they show some improvements of the classification while using parameter reduction.

4.2 Classification of eDonkey

In fact, the classification accuracy of eDonkey reached in most cases values of only about 25%, and in some cases up to 80%. The reduction of the statistical parameters could increase the classification for all protocols, but the improvement of eDonkey was significant. Fig. 1 shows the results of the classification for eDonkey with all combinations of the six parameter classes. Every combination consists of one or more parameter classes, and every parameter class is referenced by a number (see Fig. 1). The sum of the reference numbers is assigned to those combinations that consist of more than one class.

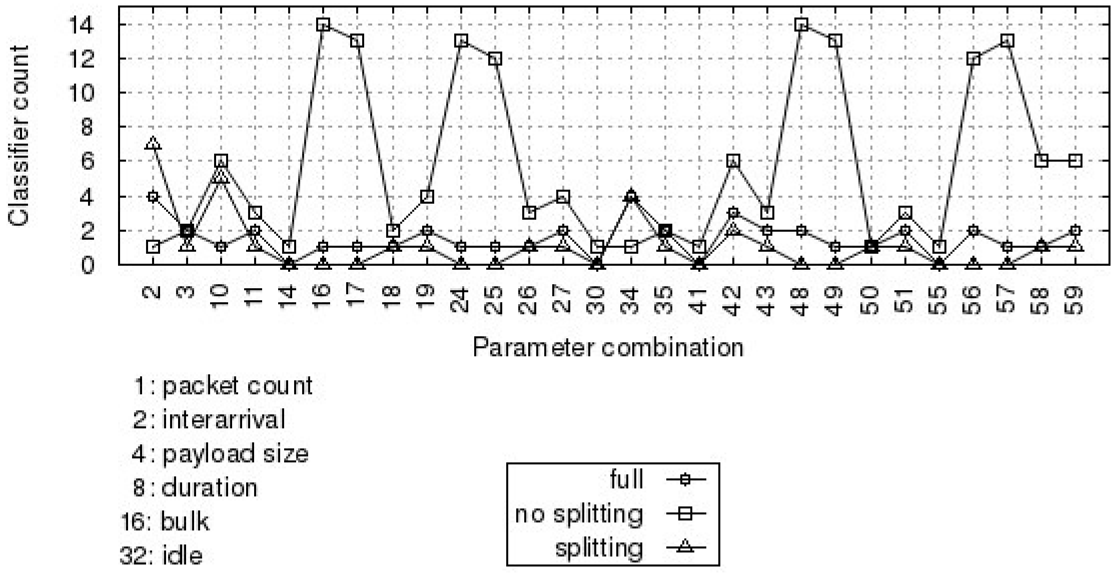

FIGURE 1: The number of classifiers that match eDonkey with a particular parameter combination with an accuracy of 90%.

Fig. 1 shows the number of classifiers that reach an accuracy of 90% or more for the different parameter combinations. We can see that some parameter combinations of the parameter category “no splitting” gain significantly more classification accuracy than the other parameter combinations. These parameter combinations are 16 (bulk), 17 (bulk + packet count), 24 (bulk, duration), 25 (bulk, duration, packet count), 48 (bulk, idle), 49 (bulk, idle, packet count), 56 (bulk, idle, duration) and 57 (bulk, idle, duration, packet count).

It is evident that the classes interarrival and payload size are not in any parameter combination that has a high accuracy. This can be explained with the nature of eDonkey's Peer-to-Peer (P2P) protocol behaviour. The eDonkey network packets differ in size because the configuration and management packets contain less data, whereas packets for data transfer can contain more data. Due to the P2P behaviour of eDonkey, many connections to peers spread over the whole world can be established. Thus, the parameter classes payload size and interarrival are not good criteria for eDonkey classification.

In contrast, the parameter class “bulk” is very important for detecting eDonkey. All parameter combinations with high accuracy contain bulk. The bulk transfer mode is a typical characteristic of data transfer protocols without application level acknowledgements. The parameter ``bulk" may also be able to increase the detection accuracy for other protocols used for data transfer like FTP, HTTP, other P2P file-sharing protocols or file-sharing integrated in instant messaging or VoIP protocols/applications.

4.3 Testing effort

As seen in Table 2, the time needed for training and testing seems to be correlated with the number of parameters used for classification. In this section, we want to have a closer look at the connection of runtime and the number of parameters. The diagrams of Fig. 2, Fig. 3 and Fig. 4 show the runtime of the different classifiers and the number of parameters used for testing. The timing results were split into three diagrams because of different scaling and different correlations between parameter count and runtime.

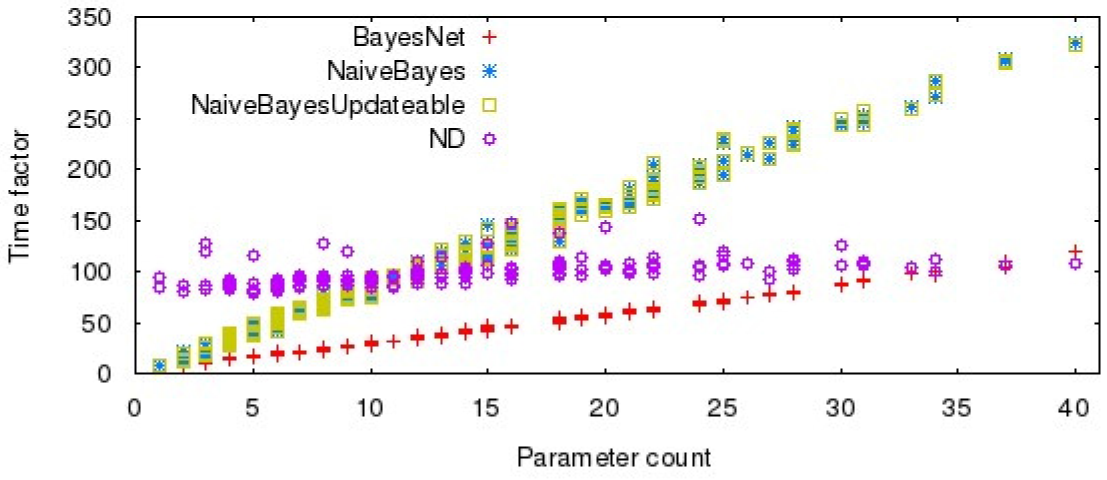

FIGURE 2: Linear time factor growth of slow classifiers.

Fig. 2 shows the runtime of the classifiers BayesNet, NaiveBayes, NaiveBayesUpdateable and ND. These classifiers are the slowest of all determined classifiers. All four classifiers rise linearly with growing parameter count. The increase of runtime for the three Bayes classifiers is significantly higher than on all other classifiers. ND has an increase similar to the classifiers in Fig. 3, but it has a huge offset. That means ND is per se very slow – independent of the parameter count.

The classifiers in Fig. 3 also increase linearly too, but they grow much slower. OneR, RandomForest and REPTree are the fastest classifiers with lowest increase. The classifiers in Fig. 4 do not show a linear increase as well. The parameter count is not the only factor for the growing runtime. The lower and upper bounds of the scatter plots in Fig. 4 are linearly increasing.

As a result, we can say that the runtime for the most classifiers is correlated to the amount of parameters. So, the reduction of parameters – with equal classification accuracy – is desirable. The Bayes classifiers are very slow and should only be used with combinations having few parameters. A prediction of the runtime of the classifiers of Fig. 4 is not possible, but the lower and upper bounds of the scatter plots give an indication. Furthermore, the ND classifier is very slow in all cases and is not suitable for real-time classification.

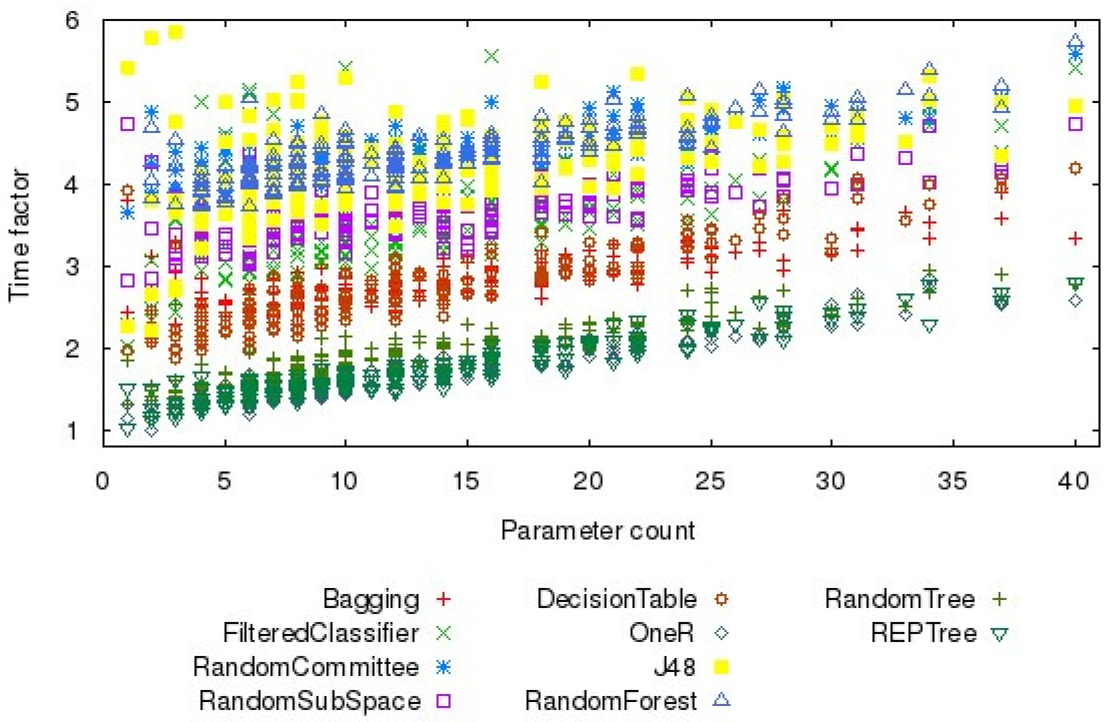

FIGURE 3: Linear time factor growth of fast classifiers.

5. Related Work

As described in the introduction, the machine learning approach has been discussed in numerous papers [3, 4, 5, 6, 7, 8] before. In this section we want to compare our findings with the results of previous papers and expose the differences. First, all previous papers used only one algorithm to classify the traffic. In some papers, however, different methods to improve the algorithms were used additionally [7, 8].

A big issue in all related work is the data pool of network traffic. There are no up to date full payload internet traffic traces available. All public available traces are truncated after the transport layer header. So, a responsible estimation of the included payload and the used upper layer protocols is not possible. Therefore, port numbers are often used to classify the truncated packets [6]. For this paper we classified and verified all traffic flows by hand to give the machine learning algorithms reliable information for the learning phase. In addition, the knowledge of the available protocols and applications is important for validating the results.

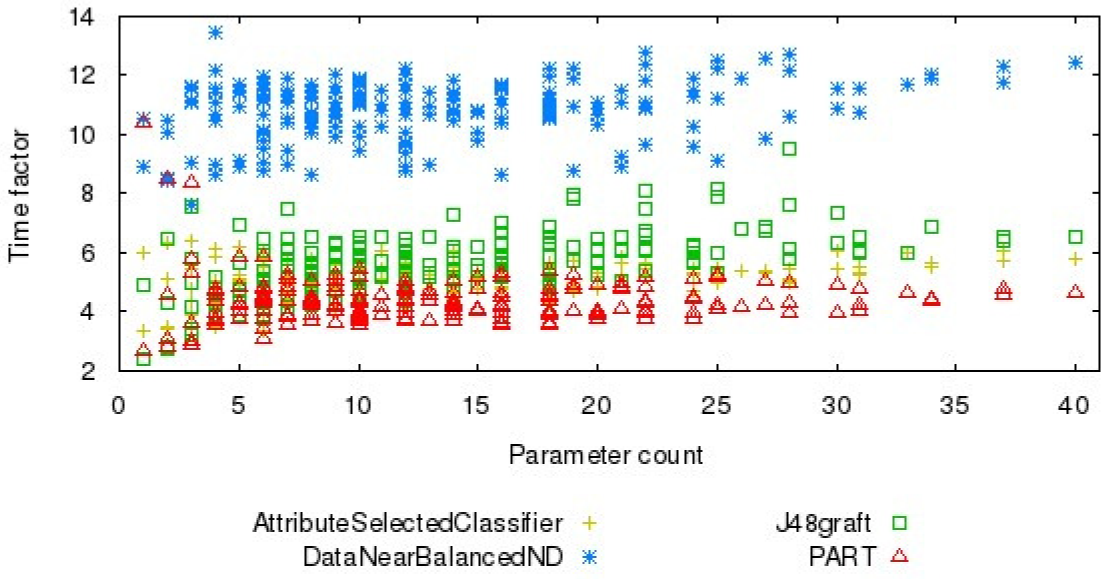

FIGURE 4: Classifiers without determinable time factor behaviour.

Another key issue is the choice of network protocols to investigate. All studies – this study included – are using only a small amount of about ten protocols or applications. For real use the 50 most used protocols should be observed. This issue is mostly due to the lack of available network traces. But this can sophisticate the results. If we use only very heterogeneous protocols, the classification is, on the one hand, easier and the machine learning algorithms will work more responsibly; on the other hand, the *false positive* rate is much lower. This effect can be seen in the results of Table 1. The protocol IMAP, for example, was falsely predicted as another protocol. The same behaviour can be seen in paper [3]. In this paper the detection rate (*true positive*) is 80% or more. However, the protocol POP has a classification accuracy of 0%, because it is very similar to NNTP and SMTP which results in false classification (false positive). Because of the small number of investigated protocols – in all studies – the effect of false positives is not considered sufficiently.

Besides the 18 machine learning algorithms, we investigated 40 statistical parameters to determine their influence on traffic classification. We only used statistical parameters, whereas other studies have also used parameters like IP addresses or port numbers [4, 8], which in effect reduces the machine learning approach to absurdity. As discussed in Section 4, the kind of statistical parameters and the number of parameters is important for the classification accuracy. This result can also be seen in [3, 6, 7]. However, this is dependent on the observed protocol and the used machine learning algorithm – a key result which cannot be read out of the other studies [3, 4, 5, 6, 7, 8].

6. CONCLUSION AND FURTHER WORK

In this paper, we have determined the impact of network protocol asymmetry according to the classification accuracy of network traffic. We used 63 combinations of six statistical parameter classes in three parameter categories consisting of up to 40 parameters. The three categories split the statistical parameters into parameters for the whole flow without differentiation of upstream and downstream, parameters that differentiate between upstream and downstream with regard to the asymmetry of some protocols, and the third category containing all 40 parameters.

Our test results show that the differentiation of the traffic can increase the classification accuracy if the parameters can expose the asymmetry of protocols like HTTP, which has an asymmetric behaviour. This asymmetry can be lost if the statistical parameters are only computed for the whole flow. Furthermore, the reduction of parameters can increase the classification accuracy. This is significant for the eDonkey protocol. Removing the parameter classes “interarrival” and “payload size” enhanced the classification accuracy of 14 of the 18 classifiers to at least 90% when the parameter “bulk” was used. This indicates that the parameter ``bulk" can be used to detect asymmetric bulk data transfer.

Additionally, the parameter reduction can decrease the runtime of the classifiers. Most classifiers have a linearly increasing runtime with reference to the parameter count. This is important for real-time traffic classification. The classifier ND showed a low classification accuracy and a high runtime, so it can be dismissed for network traffic classification. The Bayes classifiers should only be used for a very small number of parameters, because their runtimes increase very fast with increasing parameter count. Besides, the NaiveBayes and NaiveBayesUpdateable classifier show a similar behaviour and classification accuracy, and therefore, in future work it is sufficient to investigate only one of these two algorithms.

6.1 Further Work

In future we want to investigate if it is possible to train the machine learning algorithms for protocol classes such as e-mail, bulk data transfer, P2P, interactive, gaming or multimedia to enable the classification of unknown protocols belonging to such a class.

The network traffic used in this study does not reflect network traffic in the Internet. Because other studies showed that the accuracy of the classification results can vary by testing network traffic from other locations [7, 14, 15, 16], we have to repeat our investigation with other network traffic to evaluate our results.

To obtain better classification results, we have to study the best suitable algorithms in detail to adapt these generic algorithms for traffic classification. Another result of this study may be an answer to the question of why some classifiers are more suitable for traffic classification than others.

Finally, to enhance the accuracy of traffic classification, we have to find other parameters. New parameters should take the payload characteristics into account. Above all, parameters which characterise payload properties of the network protocols can enhance the classification accuracy of encrypted protocols.

6.2 Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work is supported by the European Regional Development Fund (ERDF) and the Free State of Saxony.

Gefördert aus Mitteln der Europäischen Union

Europa fördert Sachsen.
EFRE
Europäischer Fonds für regionale Entwicklung

7. REFERENCES

1. “Network Based Application Recognition (NBAR),” Cisco® , Oct. 2011, http://www.cisco.com/en/US/products/ps6616/ products ios protocol group home.html.

2. "Application Layer Packet Classifier for Linux," Oct. 2011, http://l7-filter.sourceforge.net.

3. L. Bernaille et al., "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, April 2006. [Online]. Available: http://doi.acm.org/10.1145/1129582.1129589

4. H. Jiang et al., "Lightweight application classification for network management," in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, ser. INM '07. New York, NY, USA: ACM, 2007, pp. 299–304. [Online]. Available: http://doi.acm.org/10.1145/1321753.1321771

5. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE In Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

6. S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference* on, Nov. 2005, pp. 250–257.

7. P. Piskac and J. Novotny, "Using of time characteristics in data flow for traffic classification," ser. AIMS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 173–176. [Online]. Available: http://dl.acm.org/citation.cfm?id=2022216.2022243

8. A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '05. New York, NY, USA: ACM, 2005, pp. 50–60. [Online]. Available: http://doi.acm.org/10.1145/1064212.1064220

9. "TCPDUMP & LibPCAP," http://www.tcpdump.org.

10. M. Finsterbusch, C. Richter, and J.-A. M¨uller, "Parameter Estimation for Heuristic Based Internet Traffic Classification," in *ICIMP 2012: The Seventh International Conference on Internet Monitoring and Protection*, IARIA, Ed. Stuttgart, Germany: IARIA, 2012, ISBN: 978-1-61208-201-1.

11. M. Canini, W. Li, and A. W. Moore, "GTVS: boosting the collection of application traffic ground truth," University of Cambridge, Tech. Rep. UCAM-CL-TR-748, 2009. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-748.pdf

12. M. Hall et al., "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

13. K. O'Hair, "HPROF: A Heap/CPU Profiling Tool in J2SE 5.0."

14. A. W. Moore, M. L. Crogan, and D. Zuev, "Discriminators for use in flow-based classification," Queen Mary University of London, Tech. Rep., 2005.

15. A. Finamore et al., "Kiss: Stochastic packet inspection classifier for udp traffic," *Networking, IEEE/ACM Transactions* on, vol. 18, no. 5, pp. 1505 –1515, 2010.

16. C. Rotsos et al., "Probabilistic graphical models for semi-supervised traffic classification," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, ser. IWCMC '10. New York, NY, USA: ACM, 2010, pp. 752–757. [Online]. Available: http://doi.acm.org/10.1145/1815396.1815569