MASTER THESIS

Real-Time Detection of Encrypted Traffic based on Entropy Estimation

prepared for the
Salzburg University of Applied Sciences
Degree Program
Information Technology & Systems Management

submitted by:
DI (FH) Peter Dorfinger

Head of Faculty:	FH-Prof. DI Dr. Gerhard Jöchtl
Supervisor:	Mag. Dr. Peter Kritzer

Salzburg, August 2010

Affidavit

Herewith I, Peter Dorfinger, declare that I have written the present master thesis fully on my own and that I have not used any other sources apart from those given.

First Name Surname	0910581003 Registration Number

Acknowledgement

The greatest mistake you can make in life
is to be continually fearing you will make one.
Elbert G. Hubbard (1856-1915)

First, I want to thank my supervisor Peter Kritzer, who gave me valuable advices to make a high quality thesis. He managed it to challenge me throughout the thesis to perform my best.

Great thanks go to my colleagues for spending their time for fruitful discussions and thus significantly improving my work. While it is not possible to provide an exhaustive list here, I want to thank two people for their input. First Georg Panholzer who helped me to make working code out of my thoughts. Second Wolfgang John who provided me with his set of ground truth traces for my evaluation.

To family and friends I say thanks for taking me away from work, and make my life so enjoyable. Renate, Tobias and Verena managed that it does not take longer than five minutes to forget the work and enjoy life, which was a prerequisite to finish my thesis.

Details

First Name, Surname:	DI (FH) Peter Dorfinger
University:	Salzburg University of Applied Sciences
Degree Program:	Information Technology & Systems Management
Title of Thesis:	Real-Time Detection of Encrypted Traffic based on Entropy Estimation
Academic Supervisor:	Mag. Dr. Peter Kritzer

Keywords

1^{st} Keyword:	traffic classification
2^{nd} Keyword:	entropy estimation
3^{rd} Keyword:	user privacy
4^{th} Keyword:	real-time detection
5^{th} Keyword:	traffic filtering

Abstract

This thesis investigates the topic of using entropy estimation for traffic classification. A real-time encrypted traffic detector (RT-ETD) which is able to classify traffic in encrypted and unencrypted traffic is proposed. The performance of the RT-ETD is evaluated on ground truth and real network traces.

This thesis is opened by some introductory chapters on entropy, pattern recognition, user privacy and traffic classification. A real-time encrypted traffic detector which is targeted to operate in a privacy preserving environment is presented. The RT-ETD consists of several modules that can be used to customize the approach for specific needs. A customization for two different tasks is performed, where unencrypted traffic is dropped and only encrypted traffic is forwarded.

The classification of the RT-ETD is solely based on information gathered from the first packet of a flow. Header fields as well as the payload are taken into account. The core concept of the RT-ETD is based on the estimation of the entropy of the payload, and a comparison of the retrieved value to the entropy of a uniform distributed payload.

Based on ground truth traces with encrypted traffic and real network traces it is shown that the RT-ETD is able to filter out a large fraction of unencrypted traffic, whereas a large fraction of encrypted flows is forwarded. The optimal parameterisation of the RT-ETD depends on the trade-off between detection performance and privacy preservation.

Contents

List of Figures

List of Tables

1

Introduction

During the last years there has been an enormous increase in applications which are using the Internet for communication. Applications like P2P file-sharing, VoIP communication, Video on Demand or videoconferencing are of growing interest. Due to limited network resources such applications are raising new needs within the Internet. Several methods have been proposed to better support such applications, these methods are summarized under the term "Quality of Service (QoS) for the Internet". QoS supports services that have specific qualitative needs by providing them with the appropriate quality.

To provide QoS to these applications the traffic belonging to these applications has to be identified within a multiplexed traffic aggregate. This identification of traffic is known as traffic classification. Traffic classification is further needed to understand the network behaviour, which is an essential prerequisite for network operation. Based on an improved understanding, for example network protocols are enhanced or a better network management is applied.

A major trend in the Internet is that legislation evolves into digital life. During the last years the legislation has enacted a few laws which extend the right for privacy to the world of computer networks. Content providers and Internet Service Providers have to respect and protect the privacy of their customers. Consequently, new architectures and algorithms have to be developed to support network operation in accordance with user privacy. Especially the research field of passive network monitoring is influenced by these laws, as traffic from real users is collected in passive monitoring. Legacy network

monitoring applications are reading all customer traffic to perform their algorithms. These applications have to be modified to perform theirs tasks on a pre-filtered subset of the original traffic. The pre-filtering has to be applied as "close" to the collecting point as possible.

This thesis focuses on a traffic classifier that is capable of classifying traffic into encrypted and unencrypted traffic. The classifier is used to drop unencrypted traffic which ensures that only encrypted traffic can be further processed. This approach supports network operators in a privacy preserving network operation. As only the first packet of a flow is processed the algorithm is capable to operate in a real-time environment. The approach is called real-time encrypted traffic detector (RT-ETD). The core concept of the RT-ETD is based on estimating the entropy of the payload.

It is shown that the proposed approach is able to identify encrypted flows and operate in an online/live/real-time environment as a pre-filter for advanced classification approaches.

The remainder of this thesis is organized as follows:

Chapter 2 gives a detailed overview on entropy. The chapter further presents different methods for estimating the entropy. The focus of the chapter is on entropy estimation in an undersampled environment as this is needed as an essential prerequisite for the RT-ETD. The usage of entropy estimation as a basis for a test on uniformity is presented as well as alternatives to entropy for tests on uniformity are provided.

Chapter 3 briefly introduces pattern recognition/classification. The main steps needed before classification can be performed are presented. The challenges in pattern classification are discussed based on an example. The integration of a priori knowledge into classification is shown by the usage of the Bayes formula.

Chapter 4 considers approaches for traffic classification. Starting from historical port-based classification methods and evolving towards recent algorithms. Major challenges in traffic classification are discussed. A focus is on algorithms that are able to classify encrypted traffic and on the usage of entropy in traffic classification.

Chapter 5 presents potential conflicts between network operation and user privacy. A framework for privacy preserving network monitoring, where the RT-ETD can be

used, is described. It further provides information about the conflicting situation of content providers and user privacy.

Chapter 6 investigates a new approach on real-time detection of encrypted traffic. Several modules that can be used in a real-time detector are described, where the focus is on the usage of entropy on the payload of the first packet. The algorithm is further customized as a pre-filter for two different traffic classifiers.

Chapter 7 reports evaluation results for the approach using ground truth traces and real network traces. The influence of setting two parameters is evaluated. The applicability of the approach as a pre-filter for two different types of traffic classifiers is demonstrated.

Chapter 8 concludes the presented work and indicates directions for future work.

Further all abbreviations that are used within this work can be found in the List of Abbreviations, at the end of the thesis.

2

Entropy and Entropy Estimation

This chapter provides a description of *entropy*. Entropy is a quantity that can be used to evaluate the information content of a message. The chapter will further evolve into entropy estimation and entropy estimation in an undersampled environment as these are core concepts of the presented work. The chapter is based on [11, 24, 44, 70].

2.1 Entropy

In information theory, it is a goal to transmit information with as few resources as possible. It is favourable to transmit a message with 4 bits rather than 5 bits if the same information can be represented by 4 bits. The main issue is to use statistical knowledge about the data to be transmitted to reduce the required capacity of the channel for transmission. This is ensured by a proper encoding of the information. At this point entropy comes into play. Entropy is used as a kind of measure to represent the information content (number of bits for transmission), based on statistical knowledge of the transmission content.

Suppose that we would like to transmit the winner of a horse race with eight horses. The simplest way would be to give each horse a unique number starting from 000 to 111 in binary representation. Transmitting the winner of the horse race would need 3 bits per race. This is a primitive variant, but if we assume that the probabilities of winning for the eight horses are $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{64}$, $\frac{1}{64}$, $\frac{1}{64}$, $\frac{1}{64}$, it is obvious that we are transmitting the sequence 000 (of the first horse) more often than those of the other

horses. More precisely, the sequence 000 will be transmitted for 50 percent of the horse races. It is obvious that using a shorter sequence for the first horse and longer numbers for Horses 5 to 8 will potentially have the effect that for a number of races fewer bits have to be transmitted than in the simple case. We will use this example repeatedly within this section.

In 1948 Shannon [70] developed a measure for the uncertainty of a message. This measure is known as entropy in information theory. Shannon considers the case where we have a fixed number m of possible events $A_1, ..., A_m$ whose probabilities of occurrence $p_1, p_2, ..., p_m$ are known. For the horse race example we have $m = 8$, the events are "`Horse 1 wins`", ..., "`Horse 8 wins`" with the probabilities given above. This is all we know about which event will occur. Based on the probabilities, we would like to define how uncertain we are about the outcome. The uncertainty depends on how likely it is that an event occurs. In the horse race example the uncertainty would be higher if all horses had the same probability of winning the race. Shannon defines a measure $H(p_1, p_2, ..., p_m)$ of how uncertain we are about the result of an experiment. Shannon states three desirable properties for such a measure.

1. "H should be continuous in the p_i" [70, P. 10], $1 \leq i \leq m$.

2. "If all the p_i are equal, $p_i = 1/n$, then H should be a monotonic increasing function of n. With equally likely events there is more choice, or uncertainty, when there are more possible events." [70, P. 10]. Our notation for n is m.

3. "If a choice be broken down into two successive choices, the original H should be the weighted sum of the individual values of H" [70, P. 10]. The meaning of this for the horse race example, having five successive choices, is illustrated in Figure 2.1. The final probabilities are the same in the left part of the figure as in the right part. To have a measure $H(p_1, p_2, ..., p_m)$ that fulfils this property we require, that, in our example

$$\begin{aligned} H\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\right) &= H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{2}H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{4}H\left(\frac{1}{2}, \frac{1}{2}\right) + \\ &\quad + \frac{1}{8}H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{16}H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right). \end{aligned} \quad (2.1)$$

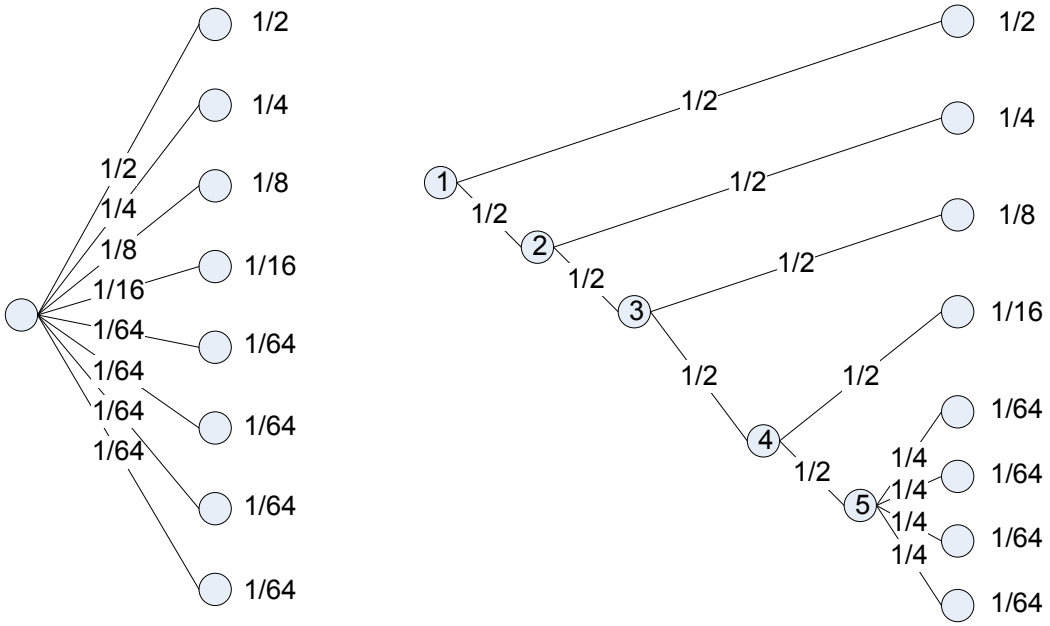


Figure 2.1: Decomposition of the choice for horse race example

In our example the choice is broken down into five successive choices. Decision number two only occurs half of the time, consequently there is the coefficient of $\frac{1}{2}$ in Equation (2.1) before the second term. The last term in Equation (2.1), choice number five in Figure 2.1, only occurs in $\frac{1}{16} = \frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2}$ of the cases.

According to [70] the only H that satisfies the three assumptions above is of the form:

$$H = -K \sum_{i=1}^{m} p_i \log(p_i), \tag{2.2}$$

where K is a positive constant and log is the logarithm to the base 2, which will be used throughout the whole thesis. For entropy $K = 1$ is used.

Based on a top down view, entropy provides us a measure of how many yes/no questions we need on average to identify the actual value of a discrete random variable. Based on the horse race example, this would mean how many yes/no questions we need on average to find out the winner of the race. For the given probabilities it is intuitively clear that we first ask “Has horse number one won the race?”. In half of all the races we would only need one question to get a “yes” and we are done. If we get a “no” the second question would be “Has horse number two won the race?”. For 75 percent of the races we will get a yes with at most two questions. For the other horses we will

continue in the same way.

The probabilities are used to ask the questions in an appropriate order to get the answer with the minimum of yes/no questions. The answers of the yes/no questions can be directly related to sending the winner of the race as a bit stream. Or, in other words, each yes/no question is seen as one bit, if the answer is "yes" a 1 will be used, if "no" a 0 will be used. So we do no longer use the numbers 000 to 111 for Horse 1 to 8 for representing the number of the winning horse. If Horse 1 wins the race we will only use one yes/no question. Consequently we will only use one bit that represents the number of Horse 1 where the value of the bit is 1. If Horse 2 wins the race we will need two yes/no questions so two bits will be used, 01. For Horse 3 we ask three yes/no questions so the representation would be 001. Horse 4 will be dealt with in the same way, 0001. For the Horses 5 to 8 the probability of winning is equal thus we cannot get any advantage from asking questions in an appropriate order. Consequently, for all the remaining horses the same number of bits is the minimal choice. Horses 5 to 8 will have the bit streams 000000,000001,000010 and 000011. For the simple case the average message length was 3 bits (000 to 111). By changing the representation format we now have an average length of $34/8 = 3.25$ bit. If we would like to transmit the winner of the horse race we will have to transmit 1 bit in 50 percent of the races, 2 bits in 25 percent of the races and so on. So the expectation of the number of bits that we have to transmit will be

$$0.5{\cdot}1+0.25{\cdot}2+0.125{\cdot}3+0.0625{\cdot}4+0.015625{\cdot}6+0.015625{\cdot}6+0.015625{\cdot}6+0.015625{\cdot}6 = 2 \tag{2.3}$$

bits. For this example this is simple, but if we do not have the bit representation of a more complex scenario how should we get the average number of bits that have to be transmitted? This is where entropy is used. Entropy provides, for given probabilities, the number of bits that are needed to transmit the information (in our case the winner of the race).

For the horse race example, the entropy H is equal to two:

$$H = \frac{1}{2}\log\left(\frac{1}{2}\right) + \frac{1}{4}\log\left(\frac{1}{4}\right) + \frac{1}{8}\log\left(\frac{1}{8}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{4}{64}\log\left(\frac{1}{64}\right) = 2, \tag{2.4}$$

A_i	p_i	$h(p_i)$	A_i	p_i	$h(p_i)$	A_i	p_i	$h(p_i)$	A_i	p_i	$h(p_i)$
a	0.0575	4.1	h	0.0313	5.0	o	0.0689	3.9	v	0.0069	7.2
b	0.0128	6.3	i	0.0599	4.1	p	0.0192	5.7	w	0.0119	6.4
c	0.0263	5.2	j	0.0006	10.7	q	0.0008	10.3	x	0.0073	7.1
d	0.0285	5.1	k	0.0084	6.9	r	0.0508	4.3	y	0.0164	5.9
e	0.0913	3.5	l	0.0355	4.9	s	0.0567	4.1	z	0.0007	10.4
f	0.0173	5.9	m	0.0235	5.4	t	0.0706	3.8	-	0.1928	2.4
g	0.0133	6.2	n	0.0596	4.1	u	0.0334	4.9			

Table 2.1: Probabilities and Shannon information content for an English text, see [44]

which means that the winner of the horse race can be represented by using two bits on average. Suppose that each horse wins with an equal probability of $\frac{1}{8}$. In this case, the entropy H will be three. So transmitting the winner of the horse race with equal probabilities will be more costly, in the sense of transmitted bits, than transmitting the winner in the example above. The entropy gives us the minimum number of bits that are needed on average to transmit information. In other words, if we use more bits than specified by the entropy we are wasting resources as we do not represent the information with the minimum number of bits that can represent the same information.

According to [44] the information content h of an event A_i can be defined as

$$h(A_i) = log\left(\frac{1}{P(A_i)}\right), \tag{2.5}$$

where $P(A_i)$ is the probability for event A_i, $1 \leq i \leq m$.

We now look at a scenario that is more likely to be used in the real world than just transmitting the winner of a horse race. We still use a simplified scenario so the focus can be put on the real important aspects. Suppose the task is to transmit an English text with signs "`a-z`" and a space character "`-`". We have a total of twenty-seven letters. It is clear that "`e`" will be more likely to occur than "`z`". Table 2.1 shows the probabilities and the information content based on an average English text.

For the representation of 27 characters 5 bits are needed. So in a simple mapping each sign would have a 5 bit representation. Consequently, transmitting an English text with 1000 signs would lead to the transmission of 5000 bits. The entropy based on Table 2.1 leads to a value of 4.11 bits. So in theory there exists a representation of the 27 signs ensuring that for 1000 signs only 4110 bits have to be transmitted on average,

A_i	Code	A_i	Code	A_i	Code	A_i	Code
a	0111	h	11010	o	0110	v	1111111110
b	11111101	i	1000	p	111100	w	11111110
c	11100	j	111111111111	q	11111111110	x	1111111101
d	11011	k	1111111100	r	1011	y	111110
e	0100	l	11000	s	1010	z	111111111110
f	111101	m	11101	t	0101	-	00
g	11111100	n	1001	u	11001		

Table 2.2: Probability-based coding scheme

a reduction of almost 20 percent.

Table 2.2 shows a straightforward approach to encoding the 27 signs based on the probabilities in Table 2.1. The average number of transmitted bits per sign is 4.12 in this case. Compared to the simple 5 bit encoding the number of bits for transmission of an English text can be reduced by about 18 percent, so that for the straightforward approach we do almost reach the ideal representation which would lead to 4.11. The entropy gives us a lower bound for the length of the representation which can be achieved. Nevertheless, for reduction of transmitted data the entropy gives us valuable information about how close we are to the optimal transmission.

Entropy has a few interesting properties that make it reasonable to be used as a measure of information. The most important ones for this work are listed here.

1. $H = 0$ if all the p_i except one, which is one, are zero. In the horse race example this would be if all races are won by the same horse. We are sure who will win.

2. Based on a given m, H attains its maximum ($\log m$) if all the p_i are equal $\frac{1}{m}$. This is the case if all horses win with the same probability, so we are most uncertain who will win.

3. Changing the probabilities toward a uniform distribution increases H. If we reduce the probability for Horse 1 to $\frac{1}{4}$ and increase for Horse 3 by $\frac{1}{4}$, H will increase from 2 to 2.16.

For more properties of H, see [70].

Figure 2.2 is a plot of the entropy for an example with 2 cases occurring with probabilities p and $q = 1 - p$ as a function of p. The figure clearly shows the properties of H

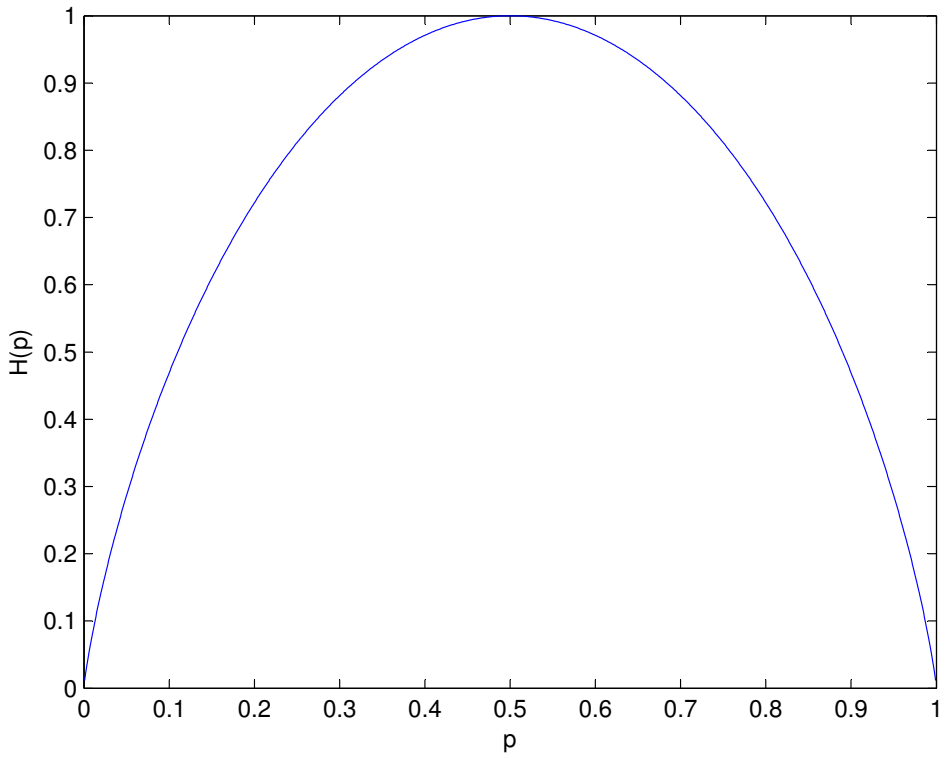


Figure 2.2: H(p) vs. p, see [70]

as stated above. The entropy is zero if $p = 0$ or $p = 1$ which means we are sure about the outcome. H increases to one at the point where $p = 0.5$ which means that both cases are equally likely. Increasing p further reduces H again.

To encode text such that only a minimum number of bits have to be transmitted can be seen as data compression. Data transmission such that the expected number of bits equals the entropy can be seen as ideal data compression. There exists no representation of the same information that can make use of less than this average number of bits. Consequently entropy can also be used as a measure of data compression.

Besides ideal transmission / data compression there is one further major domain where entropy is of great importance: encryption.

Therefore we now change the view, we do not longer try to encode data for transmission, but we try to decode a transmitted bit stream. Consequently we are now located at the receiving end of a connection.

For the example of the English text, we assume that each letter is encrypted by using a unique 5 bit representation. Someone who would like to decrypt this text being aware that the text is English has now the possibility to decrypt the text simply by using probabilities of signs in an English text. The approach to decrypt the text is straightforward. Find the bit representation with the highest probability and assume this is a "`-`". The second highest probability is assumed to be an "`e`". Proceed in the same way for a few further signs. The lowest probabilities correspond to the letters "`j`", "`q`" and "`z`". Then decrypt the text based on your assumptions, compare the words to English text and adjust your assumptions slightly to fit to English words. This is a method to decrypt a message by only using well known statistical information.

A major goal for a good encryption algorithm is to remove any predictable behaviour present in the source data, because such information can be used to fully or partially decrypt the content. Removing predictable behaviour would mean to have equal probabilities for "`0`" and "`1`" bits and even for all bit combinations like "`010`" and "`111`". Consequently a good encryption algorithm would have equal probabilities for any bit strings within the data. A data source having equal probabilities leads to high/maximum entropy. Thus entropy can be used as a measure whether different probabilities for different bit combinations can be identified. A large value for the entropy means that the used sign to bit mapping is close to the ideal mapping.

Let us focus on compressed traffic again. As an example of receiving a compressed bit stream, we assume that we are receiving the winner of 64 horse races with the probabilities for winning given in Figure 2.1. The winners are encoded by our compressed representation ("`Horse 1 wins`" equals "1", ..., "`Horse 8 wins`" equals "000011"). We now look at the probabilities for "`0`" and "`1`" bits. Horse 1 will win 32 races, consequently we will transmit 32 times 1, Horse 2 will win 16 races, consequently we have to transmit 16 times 0 and 16 times 1 (01). A full list of the number of transmitted bits per winning horse can be found in Table 2.3. The receiver will receive a bit sequence where the number of zeros and ones is equal (64 in our example). As already argued above, equal probabilities lead to maximum entropy.

Consequently if we are receiving a bit stream where the entropy is high it can be assumed that the data has gone through a data compression or an encryption process,

Horse	sequence	wins	#1	#0
Horse 1	1	32	32	0
Horse 2	01	16	16	16
Horse 3	001	8	8	16
Horse 4	0001	4	4	12
Horse 5	000000	1	0	6
Horse 6	000001	1	1	5
Horse 7	000010	1	1	5
Horse 8	000011	2	2	4
			64	64

Table 2.3: Number of transmitted bits for 64 horse races

because these two processes have the goal to encode data such that the entropy is high. The reasons to do so for the two processes are different. For data compression the goal is to reduce data to a minimum, whereas in encryption the goal is to remove any predictable behaviour that is present in the source data. So entropy can be used as a measure whether the data is encrypted/compressed or not.

2.2 Entropy Estimation

The Shannon entropy can only be calculated if the appropriate values for p_i, $1 \leq i \leq m$ are known. This is only true if the probabilities are based on an infinite amount of input data, or a closed set of data. The probabilities in Table 2.1 are based on "The Frequently Asked Questions Manual for Linux". If we would assume that this is the only document that exists in the world, the probabilities and the entropy would be accurate. If we would use a different document instead, the probabilities would be slightly different, depending on the document. Consequently the probabilities in Table 2.1 are estimations of the true probabilities, which would have to be calculated based on all documents that exist in the world. This would not be infinite but the closed set as stated above. Clearly it is not feasible to calculate the true entropy based on all documents in the world. This is the same for a wide range of application areas where entropy is used. Furthermore there is often only a sample available, in the above case only a specific document, which is used as a basis for calculating the entropy. Consequently entropy estimation plays an important role.

In the horse race example entropy estimation is simply estimating the entropy based on

Horse	wins	p_i	$h(A_i)$
Horse 1	13	0.43	1.21
Horse 2	6	0.20	2.32
Horse 3	4	0.13	2.91
Horse 4	2	0.07	3.91
Horse 5	1	0.03	4.91
Horse 6	1	0.03	4.91
Horse 7	1	0.03	4.91
Horse 8	2	0.07	3.91

Table 2.4: Probabilities and information content for a sequence of horse races

the results of a sequence of horse races. Suppose we have the winners `H1-H2-H1-H2-H1-H8-H7-H3-H6-H1-H5-H1-H4-H3-H3-H1-H2-H8-H1-H1-H2-H1-H4-H1-H1-H1-H2-H1-H2-H3` in a sequence of horse races. The sequence was produced by a random number generator using the probabilities from Section 2.1. The probabilities and the information content of each event based on these results are shown in Table 2.4. It is obvious that for different sets of races we will get different probabilities as long as the sequence is not infinite. The entropy based on the estimated probabilities is 2.39 instead of 2 for the real probabilities. Producing a further random sequence of horse races with the same length will lead to different probabilities and thus to a different value for entropy. It is obvious that the longer the sequence of races is, the more accurate the entropy estimation will be.

Mathematically formulated based on [2] the estimation of the entropy is:

"Suppose P is an arbitrary discrete distribution on a countable alphabet A. Given an i.i.d. sample (X_1,...,X_N) drawn from P, we consider the problem of estimating the entropy $H(P)$ or some other functional $F = F(P)$ of the unknown distribution P." [2, P. 1]

This is exactly what we have done above for the horse race example. N is the length of the sample, in our example the number of horse races, and an i.i.d sample is an independent and identically distributed sample.

As shown above in the horse race example there is a relevant difference between the entropy and the estimated entropy. As this is also true in general there has been lot of research on entropy estimation. Different estimators were suggested and the most important ones for our work are now presented here, see [57] for more details.

The goal of entropy estimation is to make the difference between the true entropy and the estimated entropy small independently of the distribution p. The simplest estimator is the maximum likelihood (ML) estimator [2, 72] which is also called the "plug-in" or "naive" estimator. The approach is straightforward: we count the occurrences n_i , $1 \leq i \leq m$ of each event A_i, $1 \leq i \leq m$. The frequency f_i of A_i in the sample is then n_i/N. The entropy of the sample based on the ML estimator is:

$$\hat{H}_{\text{MLE}} = -\sum_{i=1}^{m} f_i \log(f_i) \tag{2.6}$$

This leads to a systematic underestimation of the entropy [57],

$$E_p(\hat{H}_{\text{MLE}}) \leq H(p), \tag{2.7}$$

where $E_p(.)$ denotes the conditional expectation given p. Based on the above estimator several variants have been proposed with the goal to reduce this underestimation. Miller-Madow [48] introduced a bias correction to the maximum likelihood estimator (MLE)

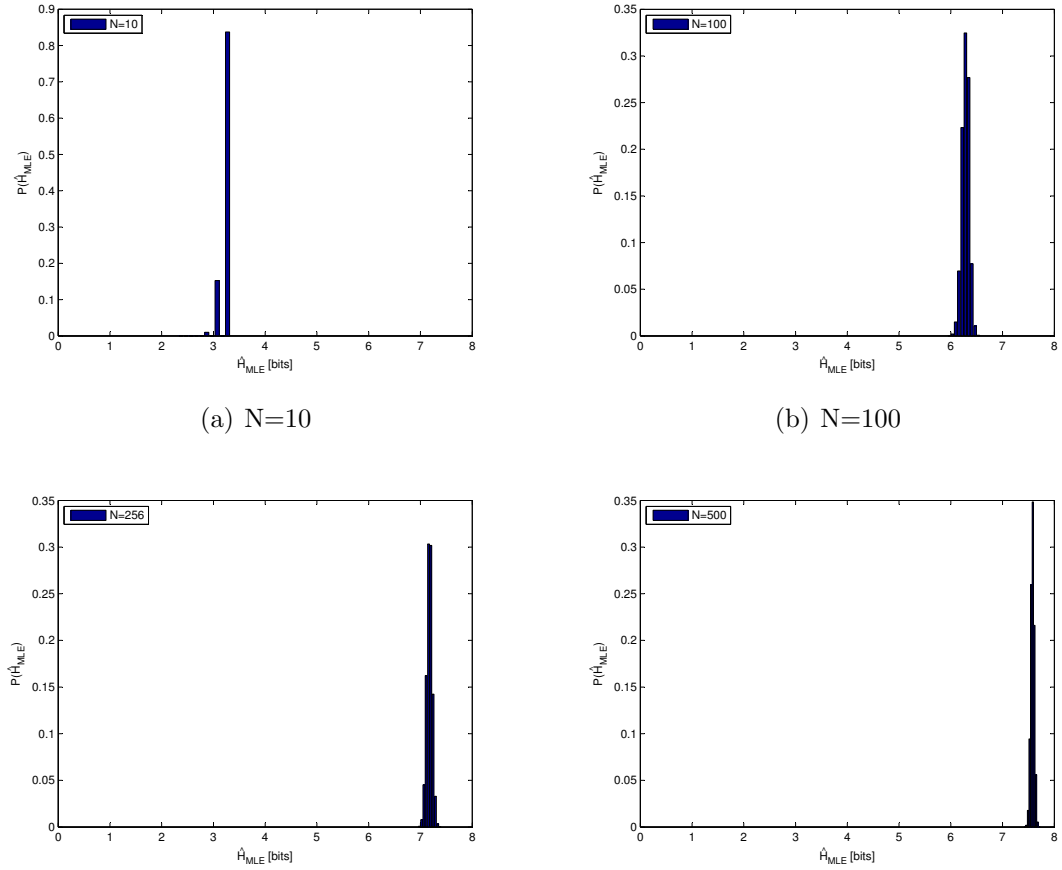
$$\hat{H}_{\text{MM}} = \hat{H}_{\text{MLE}} + \frac{M-1}{2N}, \tag{2.8}$$

where M is the number of bins with nonzero probability. A third version of the MLE is the jackknifed version [19],

$$\hat{H}_{\text{JK}} = N \cdot \hat{H}_{\text{MLE}} - \frac{N-1}{N} \sum_{j=1}^{N} \hat{H}_{\text{MLE}-j}, \tag{2.9}$$

where $\hat{H}_{\text{MLE}-j}$ is the maximum likelihood estimation based on all but the j-th sample.

Suppose we have a uniform distribution with $m = 256$ and use the MLE for different N. We performed a Monte-Carlo experiment producing 100.000 sequences of length N. The entropy was estimated using the MLE. Figure 2.3 plots the histograms of the Monte-Carlo experiments. On the x axis the estimated entropy is plotted, on the y axis $P(\hat{H}_{\text{MLE}})$ which is the probability of $\hat{H}_{\text{MLE}}$. The true value of H is eight. Figure 2.3



(a) N=10

(b) N=100

(c) N=256

(d) N=500

Figure 2.3: MLE distributions for $m = 256$ and different N

shows the most basic fact about entropy estimation: the variance is small and the bias is large until $N \gg m$. This statement holds even for the above estimators that are tailored to bias reduction [57]. Given a fixed m the estimation of the entropy is closer to the true value the greater N is.

In [57] Paninski performs a detailed analysis of the bias. He proposes an estimator based on Bernstein approximating polynomials, which are defined as linear combination of binomial polynomials. This estimator allows the experimenter to decide whether bias reduction or variance reduction is of greater importance. The Paninski estimator still leaves open problems, which prevent the estimator from being the ultimate estimator. Schuermann gives the following statement about the Paninski estimator:

> *"Unfortunately, the good approximation properties of this estimator are a result of a delicate balancing of large, oscillating coefficients, and the variance of the corresponding estimator turns out to be very large [57]. Thus,*

> *to find a good estimator, one has to minimize bounds on bias and variance simultaneously. The result is a regularized least-squares problem, whose closed-form solution is well known. However, one can only hope that the solution of the regularized problem implies a good polynomial approximation of the entropy function. The latter also depends on whether the experimenter is more interested in reducing bias than variance, or vice versa."* [68, P. L296]

Consequently, the practical usage of the Paninski estimator is limited.

2.3 Undersampled Estimation

The difficulty of estimating entropy directly depends on the quotient of the number of samples available and the number of events N/m. The higher the value of N/m, the easier is the estimation problem. The smaller the ratio, the harder it is. Especially entropy estimation in the range where $N < m$ is not a trivial task. Paninski [58] shows that it is possible to accurately estimate the entropy on m bins, given N samples, even when N/m is small. It is even possible to estimate the entropy on m bins when you have less than m samples available. But as stated above, the usage of the Paninski estimator is limited due to its complexity.

Suppose we have a sequence of eight horse races of eight horses, which means $N = m$, where the winners are based on the probabilities given in Section 2.1. We are using the Matlab [76] random number generator (`rand`) to generate uniformly distributed numbers in the half open unit interval ($[0, 1[$), where values $[0, 0.5[$ are mapped to "`Horse 1 wins`", $[0.5, 0.75[$ to "`Horse 2 wins`" and so on. We performed a run of length 8 with the random number generator and got the values [0.7180,0.1889, 0.4339,0.0456, 0.6555, 0.7810, 0.8629,0.7854]. Consequently, the winners for the races are `H2-H1-H1-H1-H2-H3-H3-H3`. Based on this sample we now estimate the entropy using the algorithms presented above. The results are given in Table 2.5.

For the Maximum Likelihood, the Miller-Madow and the jackknifed estimator the results are as expected. The Maximum Likelihood estimator has the greatest negative bias and the jackknifed the smallest. The Miller-Madow estimator is in the middle. For all three estimators the bias is negative. For the Paninski estimator the code provided

Estimator	H
Maximum Likelihood	1.5613
Miller-Madow	1.6863
jackknifed	1.9409
Paninski	1.4861

Table 2.5: Undersampled Entropy Estimation for a sequence of horse races

in [56] was used. The bias is greater than expected, the major reason being is that the parameter set for the estimator is not optimized for our use case. To optimize the parameter set is not a trivial task and, as not a major need for our work, it was not performed.

2.4 Entropy-based Test for Uniformity

A key characteristic of entropy is that if events are more equally likely the entropy will be higher, see Section 2.1. Consequently entropy can be used as a measure of uniformity. Suppose we have horse races with the above given probabilities of winning. The entropy in this case is two. For horse races where all horses win with the same probability the entropy is three.

For an entropy-based test for uniformity we would need an appropriate estimator for the entropy. As stated above this is hard to retrieve, so we focus on two uniformity tests that are influenced by entropy but do not need the estimation of the entropy.

Paninski [59] analyses how many samples N are needed from a distribution p to decide whether p is uniform or not. In the range where N is much grater than m, the answer regarding uniformity can be given by the chi-square theory. For $m \gg N$ Paninski relates this problem to the birthday problem [15] and uses the so-called "birthday inequality", which states that for uniformly distributed birthdays coincident birthdays are least likely than for any other distribution. The major finding of [59] is that for a uniformity test $N \gg \sqrt{m}$ samples are needed; fewer samples will have the problem that for at least one distribution the non uniformity will not be detected, see [59] for details.

A second approach is presented by Olivain and Goubault-Larrecq in [55] where, motivated by the problems of estimating the entropy based on a sample of length N accordingly, the N-truncated entropy is used instead. The N-truncated entropy $H_N(p)$ is defined as follows. Generate all possible words w of length N according to p. Estimate the entropy based on maximum likelihood for all words w. The N-truncated entropy is then the average of the MLE estimates, i.e. sum up all MLE estimates and divide by the number of words to retrieve $H_N(p)$.

According to [55] $H_N(p)$ can be calculated by direct summation,

$$H_N(p) = \sum_{n_1+...+n_m=N} \left[\binom{N}{n_1 + ... + n_m} p_1^{n_1} ... p_m^{n_m} \times \left(\sum_{i=1}^{m} -\frac{n_i}{N} \log \frac{n_i}{N} \right) \right], \tag{2.10}$$

where

$$\binom{N}{n_1 + ... + n_m} = \frac{N!}{n_1! ... n_m!} \tag{2.11}$$

is the multinomial coefficient.

Given that $p_i = 1/m$ for all i, which means that p is the uniform distribution U, equation 2.10 can be simplified to

$$H_N(U) = \frac{1}{m^N} \sum_{n_1+...+n_m=N} \left[\binom{N}{n_1 + ... + n_m} \times \left(\sum_{i=1}^{m} -\frac{n_i}{N} \log \frac{n_i}{N} \right) \right]. \tag{2.12}$$

The maximum likelihood estimator can then be used as an unbiased estimator of H_N. Checking for uniformity is then straightforward. Based on a sample of length N estimate the entropy using MLE and compare the result to $H_N(U)$. The closer the estimated value is to $H_N(U)$ the more likely is it that the underlying distribution is uniform. This can be seen as special case of the Kullback-Leibler distance [11]. The Kullback-Leibler distance is used to compute the divergence from one distribution to another one. In our case the reference distribution is the uniform distribution.

Based on the example in Section 2.3 where we have a sample of size 8, $H_N(U)$ is equal to 2.246. The random number generator produces the values $[0.7180, 0.1889, 0.4339,$

$0.0456, 0.6555, 0.7810, 0.8629, 0.7854]$. Consequently, based on the probabilities given in Section 2.1 the winners for the races are `H2-H1-H1-H1-H2-H3-H3-H3`. The entropy estimated by MLE is 1.5613. For the same random numbers, given equal probabilities, [0,0.125[is mapped to "`Horse 1 wins`", [0.125,0.25[is mapped to "`Horse 2 wins`" and so on, the sequence is `H6-H2-H4-H1-H6-H7-H7-H7`. The estimated entropy using MLE is 2.1556 which is closer to the truncated entropy of a uniform distribution which is 2.246, for this example. We conclude that it is more likely that the sequence `H6-H2-H4-H1-H6-H7-H7-H7` is based on a uniform distribution than the sequence `H2-H1-H1-H1-H2-H3-H3-H3`. What is still an open question is: "With which probability is one of these two sequences based on a uniform distribution?"

To answer this question we have to take a look on the confidence intervals. The second open point is the calculation of $H_N(U)$ which is getting unmanageable. The authors of [55] present solutions for an approximation of $H_N(U)$ and for estimating the confidence intervals.

The N-truncated entropy of a uniform distribution can be calculated by

$$H_N(U) = \log m + \log c - e^{-c} \sum_{j=1}^{+\infty} \frac{c^{j-1}}{(j-1)!} \log j + o(1), \tag{2.13}$$

where $c = \frac{N}{m}$ and $o(1)$ is an error term. An example plot of this error term, for $m = 256$, can be found in [55]. For $m = 256$ the error term is never more than 0.004 bits, consequently Olivain and Goubault-Larrecq state that

$$H_N(U) \approx \log m + \log c - e^{-c} \sum_{j=1}^{+\infty} \frac{c^{j-1}}{(j-1)!} \log j \tag{2.14}$$

is a good approximation of $H_N(U)$.

Details about the estimation of the confidence intervals are presented in Chapter 6.

2.5 Alternative Tests for Uniformity

Beside entropy there exist several tests for uniformity. For a selected subset of them, details will be presented in this subsection. Most of these tests are used in cryptography [47].

The chi-square test can be used to determine whether an observed frequency distribution differs from a theoretical distribution or not. If the theoretical distribution is the uniform distribution, the test can be used as test for uniformity. This test is for example used for Skype detection [5]. More details about the chi-square test can be found in [47].

Maurer's test uses the fact that it is not possible to significantly compress a uniformly distributed sequence without loss of information. As compressing the sequence is to time consuming, Maurer's test instead computes a quantity which is related to the length of the compressed sequence. More details about the Maurer test can be found in [47].

The index of coincidence I_c of Friedman [22] determines how likely it is, if we take two elements out of a sequence, that they are identical. Based on the above example of the English text (see Section 2.1), one would take two signs out of a text sequence and determine how likely it is that they are identical. For a uniform distribution the value of I_c is lower than for any other distribution. The index of coincidence can be calculated by the following equation:

$$I_c = \frac{\sum_{i=1}^{m} f_i(f_i - 1)}{N(N-1)} \tag{2.15}$$

The expectation of the index of coincidence $E(I_c)$ can be calculated by the following equation:

$$E(I_c) = \sum_{i=1}^{m} p_i^2 \tag{2.16}$$

Based on the above example of the English text $E(I_c)$ is equal to 0.0783. If we assume a uniform distribution with the same length of the alphabet $m = 27$, "`a-z`" and "`-`",

$E(I_c)$ is 0.037. Note that the factor between the expectation of the index of coincidence of a uniform distribution and the expectation of a natural language is in the range of two.

The Kolmogorov complexity [11] of an object, such as a piece of text, is a kind of measure of the minimum length of a program that is needed to reproduce this piece of text. For example the sequence "`abababababababababababab`" which can be expressed by "print 12 times ab", has a lower Kolmogorov complexity than the sequence "`a1rz6e2i8tqw03unmk79hs5o`" where the only possibility to reproduce the sequence is to print the full sequence. Since a uniform distribution produces sequences that are hard to be expressed by a shorter representation than printing the full sequence, the Kolmogorov complexity can be used as an indication for uniformity.

3

Pattern Recognition

The RT-ETD is a traffic classifier with two classes, encrypted traffic and unencrypted traffic. The core basics of traffic classification are in pattern recognition. Therefore we present the core concepts of pattern recognition.

Humans are almost perfect engines for pattern recognition and classification. For us it is quite simple to recognize a face, read handwritten characters, understand spoken language even with accent or decide whether a meal is tasty or not. All these aspects belong to pattern recognition and classification. In pattern classification a main part is the categorization into pre-defined classes. In the case where we see a new human face we would almost automatically classify it into "beautiful" or "not beautiful". In the case of handwritten characters we would classify them according to a character which is stored in our brain and looks similar to the character under investigation.

For humans pattern recognition and classification is an easy task, as we have been trained for millions of years as this has been crucial for our survival. The ability to take raw data and execute a fast action based on patterns extracted from the raw data is a key task for humans. The findings within this chapter how such tasks can be performed by computer systems are based on [18].

Compared to humans, computer systems are at the very beginning of pattern recognition and classification. A few terms of pattern recognition are used throughout this thesis and will now be defined based on the example of reading handwritten characters. A feature is a characteristic that can be extracted from a character. Features can for example be the height, width, height divided by the width, fraction of straight

lines compared to curves or angle of the longest straight line. A pattern would be the description of one character based on a set of features. This description is so specific that different patterns can be distinguished from each other. A pattern is a description of a letter that allows differentiating it from any other letter.

Pattern classification can be seen as process including two major steps. The first one is the pre-classification process. The second step is the process of classification.

3.1 Pre-Classification

Before a system can start to classify, a few steps have to be performed in advance, where the two most important ones are:

1. Feature Extraction,
2. Learning.

As a first step one has to extract features that allow to uniquely describing the patterns. In the case of handwritten characters this would allow to uniquely describe each letter based on this set of features.

The major task in feature extraction is to find the right set of features. Questions like "How much features should be used?" or "Which features are the most promising ones?" seem to be trivial but they are not. Using many features leads to highly complex patterns and tends to be too specific, which means that slight variations would not be detected. Too few features would lead to a worse differentiation between the patterns. Choosing the wrong features will result in increased complexity without increased performance.

After the selection of the appropriate features, a histogram or density function for each character and each feature has to be defined. There are three main methods for this:

1. Supervised Learning
2. Unsupervised Learning

3. Reinforcement Learning

In supervised learning, a teacher provides the appropriate character for each handwritten character in a training set. After the training phase, the classifier is evaluated using a test data set. The test data set ensures that the classifier is able to generalize. The ability to accurately classify the training set but failing with the test data set is known as overfitting, in which case the classifier is not able to accurately classify slight differences compared to the training data set which are present in the test data set and will probably be available in real data.

In unsupervised learning, also called clustering, there is no pre-classified data set. The system performs a grouping. The main input parameter is the number of clusters that the system should use.

The best way for training a classifier would be reinforcement learning, where the character is classified based on raw data. As in supervised learning the target character has to be known. After the classification the classifier will get the feedback whether the classification was appropriate or not.

A set of appropriate values of all used features that represent a pattern is also called signature. This term is often used in traffic classification.

3.2 Classification

The second step in pattern classification is the process of classification. Classification is the extraction of the above defined features from raw data followed by the decision, based on the values of the appropriate features, which character is present in the raw data. Based on raw data a few problems can emerge. Consider the hand written text in Figure 3.1. First of all the text is not horizontal, which has to be taken into account. For the first letter it is not possible to determine the feature height as parts of the line are missing. A further challenge is to determine where one letter ends and the next one begins. Especially between Letters three and four this is hard. For the last letter it is hard to determine if it is a “c” or an “e”. A human reader, who assumes that there

Figure 3.1: Example of handwritten text

are no missing letters, automatically recognizes it as an "e". What the human reader uses here is called context. The context is information gathered not from the raw data of the character itself but in this case the information of the characters before. As the word automatc does not exist, the last letter has to be an "e". What else can be used here is the fact that the letter "e" is much more likely to occur in English than "c" so also this fact can be used in such a case.

If the feature values extracted from the raw data of a character would have exactly the same signature as a pattern it would be easy to find the appropriate letter. In case of the fact that some of the values are apart, the closest pattern should be chosen. But what does "closest" mean in this case? For each feature value a probability of each pattern has to be defined. For example if the height to width ratio is two, it is much more likely that the character under investigation is a "b" than an "a". The decision should not be based on this information only. A rule in pattern recognition is to use as much pre-defined knowledge as possible for classification, the main reason being that with pre-defined knowledge the number of features that have to be extracted can be reduced.

Suppose we would like to classify the last character in Figure 3.1 without any usage of information gathered from previously read characters. We assume that, based on our features, we calculate the probability that this is a "c" with a value of 0.6, that it is an "e" with a value of 0.4. For classification we would like to use a priori knowledge of the probabilities of characters in English text. This is where the Bayes decision theory comes into play.

The a priori probability $P(e) = 0.0913$ and $P(c) = 0.0263$ (see Section 2.1) can be used for the decision. To combine a priori probabilities and feature based probabilities the Bayes formula is used.

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)}, \tag{3.1}$$

where w_j is a character and x is the feature value, $p(x|w_j)$ is the probability of feature value x for character w_j, $P(w_j)$ is the a priori probability of character w_j and

$$p(x) = \sum_{j=1}^{m} p(x|w_j)P(w_j) \tag{3.2}$$

where m is the number of characters.

Equation 3.1 can also be expressed informally by saying

$$posterior = \frac{likelihood \times prior}{evidence}. \tag{3.3}$$

In order to decide to which class the character belongs, the posterior probability for each class has to be calculated. The final decision should then be based on choosing the character with the greatest posterior probability. Note that the evidence can be neglected for this decision.

For the given example, where $P(e) = 0.0913$, $P(c) = 0.0263$ (see Section 2.1) and our assumed probabilities of 0.4 for an "e" and 0.6 for a "c", we can calculate the probability for "e" and "c". The probability $p(x|e)P(e)$ is 0.03652 and $p(x|c)P(c)$ is 0.01578. So it is intuitive that it should be decided that the character is an "e". In general it should be chosen the character with the greater value for the above, as the major goal in classification is to reduce the amount of misclassification to a minimum. It is shown in [18] that this decision will minimize the probability of an error. So choosing w_1 if $p(x|w_1)P(w_1) > p(x|w_2)P(w_2)$ and w_2 otherwise minimizes the probability of error.

The Bayes decision rule combines prior probabilities and likelihoods to achieve a decision with the minimum probability of error. So in all scenarios where the amount of misclassification should be reduced to the minimum, the Bayes formula is of importance.

For our given task to classify traffic in encrypted and unencrypted traffic the Bayes decision rule can be used for ports greater than 1023 where the fraction of encrypted traffic on the port under investigation is used as a priori knowledge. This task will be investigated in future work.

Above we assumed that the error for misclassification is equal costly, which means the risks for making the wrong decision are equal. For the given example to classify an "e" as a "c" has the same risk as to classify a "c" as an "e". Risk is a function or value for the consequences of a misclassification. In a different classification problem, where e.g., based on a scan of a fingerprint access should be granted or not, it is obvious that risks are not equal.

The risk of granting access to someone who should not have access is much higher than the risk of rejecting someone that should have access. Details about Bayes decision, where the risk should be minimized instead of the error can be found in [18].

Also in our application domain, the risk of forwarding unencrypted traffic can be higher, as it violates user privacy, than as dropping encrypted traffic. The appropriate setting for this depends on the application that uses our pre-filtered data.

4

Traffic Classification

Traffic Classification is the decomposition of network data based information, gathered from traffic on network links. The goal of traffic classification is to identify the type of traffic carried on links [6, 30, 36, 54]. Classification is performed according to application protocols or to a class of applications (e.g. bulk, interactive, WWW, ... [52]). Protocols that are often targeted by classifiers are the File Transfer Protocol (FTP) [63] which is used for file transfers, the Secure Shell (SSH) [81] protocol used for secure remote management, the Domain Name Service (DNS) [51] protocol for communication with the domain name servers which perform the mapping of domain names to Internet protocol (IP) addresses, the network management protocol SNMP (Simple Network Management Protocol) [7] or the Network Time Protocol (NTP) [49] used for time and date synchronization between computers. Additionally the most important mail protocols, the Simple Mail Transfer Protocol (SMTP) [37] used for sending emails and transmission of emails between mail servers, and the two protocols used for receiving emails POP3 (Post Office Protocol version 3) [53] and IMAP (Internet Message Access Protocol) [12], are well handled in traffic classification. And for sure the well known application protocol HTTP (Hyper Text Transfer Protocol) [21], used in web surfing, is target of traffic classification. Some of the traffic classification algorithms are able to identify even the encrypted versions of these protocols which are normally labelled by an "S", for example POP3S, HTTPS or FTPS. In recent years, peer-to-peer (P2P) communication such as e-Donkey, Kazaa or Skype, where no server is involved in communication, are responsible for a large fraction of traffic and are consequently target of traffic classification.

The results of classification are needed for a sophisticated network management. Quality of Service (QoS) mechanisms, such as token bucket mechanisms [16, 65] or weighted fair queuing [3], where the traffic is handled differently based on classification, are nowadays used in networks.

The ability to understand the type of traffic on network links can be further used to detect illicit traffic, such as network attacks and other security violations. Modern firewalls and Intrusion Detection Systems (IDSs) perform access policies based on traffic classification to protect networks from security violations. Furthermore the legal discussion about Internet usage in the field of peer to peer (P2P) file sharing between P2P communities and intellectual property representatives is a motivation for traffic classification [40].

For traffic classification, all approaches use signatures (which is a subset of features, see Chapter 3). Signatures based on different features can be very specific or wide ranging but should always ensure that the following key requirements are fulfilled [25]:

1. Accuracy, i.e. a low misclassification error rate,
2. lightweight and fast, i.e. low evaluation overhead and short evaluation time,
3. early identification of applications, i.e. right at the beginning of a connection,
4. robustness against asymmetric routing,
5. good accuracy properties over an extended period of time,
6. wide applicability, i.e. able to identify a range of different applications.

The first requirement ensures that the approach is useful at all and is, consequently, the key requirement within all approaches. Requirement 2 is motivated by the demand for online-operation on high-bandwidth links. In cases where the result of traffic classification is used to map flows to QoS classes, Requirement 3 plays a central role, as an increased QoS level should be applied right from the beginning of a connection. As network traffic may be routed on different links, traffic classification should work regardless of which direction of the traffic is used for classification. Requirement 5 should

ensure that the mechanisms work for a changing network traffic mix on the monitored link. As network load changes, relative proportions may change, which should not prevent appropriate traffic classification. If the application itself changes, signatures have to be changed for sure. Requirement 6 ensures that a major subset of the traffic on the monitored link can be classified appropriately.

Later on we will investigate the aspect that not all traffic classification approaches fulfil all the requirements, especially Requirement 4 and 6 are often neglected. In situations with symmetric routing Requirement 4 becomes unnecessary. Nowadays specifically targeted approaches exist that are only able to classify a single application (neglecting Requirement 6). However, also such approaches are of high interest, as protocols exist where it is complicated to classify them appropriately and a reasonable amount of effort is needed. Such highly specific classification approaches can be integrated into bigger classification environments. Further information about traffic classification is given in [30].

4.1 Port-based Classification

Early network applications, such as FTP, SSH or HTTP were designed to use well-known port numbers [27] for the communication between clients and servers, or between peers. These applications are either using the connection oriented bi-directional communication protocol TCP (Transmission Control Protocol) [62] or the connectionless uni-directional communication protocol UDP (User Datagram Protocol) [61]. Both protocols provide the port numbers of the connection within their headers. Consequently, traffic classification used to be straightforward. The task of traffic classification was simply to classify traffic based on port numbers in TCP/UDP headers. Table 4.1 shows a short list of the most important well known port numbers; a full list is available at [27].

During the 1990s, port-based traffic classification was the state of the art in traffic classification [67]. The major advantage of such a classification approach was a good performance to cost ratio. Traffic classification based on port numbers is implemented

Port	Application
20/21	FTP
22	SSH
23	Telnet
25	SNMP
53	DNS
80	HTTP
110	POP3
443	HTTPS
993	IMAP

Table 4.1: Important well-known port numbers

in commercially available products (e.g., Cisco NetFlow [10]). Such routers allow distinguishing the service quality, during high load, based on port numbers (e.g., given well-known ports up to 1023 higher priority than ports greater than 1023).

Up to the year 2000, port-based classification was the only method needed for a proper classification. In the early 2000s, developers of upcoming file sharing applications (e.g. Kazaa [39]) started to use dynamic port numbers. During the first years the presence of file sharing traffic was limited, thus port-based classifiers were still valid for a major part of the traffic. As file sharing applications are nowadays responsible for a major part of network traffic, the accuracy of port-based classification is limited [34, 52, 69].

Nowadays port-based traffic classification algorithms can still be used to detect legacy applications with high precision and recall. Port-based traffic classification on the full set of traffic leads to worse results, but using it on a subset of traffic the results are still satisfying. For DNS, Mail, SNMP, News and NTP port-based traffic classification approaches still lead to a classification with more than 98.9 percent precision and recall [36].

In two other areas, port-based classification fails nowadays, namely if the application is using dynamic ports, such as passive FTP download or many P2P applications. Recently developed file sharing applications and the well known VoIP Software Skype hide traffic behind well known port numbers, such as 80 or 443, to evade detecting and blocking. VoIP is short for Voice over IP where IP is the well known Internet Protocol. The overall amount of such hidden traffic and thus the influence on classification is still limited [36]. The presence of hosts using Skype increased within the last years [77], but

we are not aware of a recent work that evaluates the amount of traffic hidden within port 80 or 443. Findings from [36] might already be outdated.

4.2 Challenges in Traffic Classification

The major question that arises in this context is: "Why is traffic classification getting more and more difficult?" The answer can be given by looking at the motivation for traffic classification or more precisely application detection. This can be illustrated by a simple example such as the well known VoIP application Skype [71]. Applications such as Skype are a security risk for company networks (as they open a communication path to the Internet). Consequently, many company network administrators try to detect Skype applications and try to block the traffic. Skype developers have the contrary objective, they are implementing mechanisms that allow a widespread usage of their application. So it is more or less a race between Skype developers to hide the traffic, and traffic classification to detect the application traffic. The second major player in Skype detection is the group of Internet Service Providers (ISPs) who would like to detect the traffic on their network. ISPs often offer VoIP services and thus are interested in degrading the service quality of other VoIP services, at least at times when network load is high, but maybe also during normal operation. This potentially hurts the vision of network neutrality and on this topic there has been extensive discussion between ISPs and content providers [32, 33, 50]. Moreover, degrading service of P2P file sharing applications during high network load is of interest to ISPs, in order to allow proper operation of other services. The latter is even more important than the Skype case, as the share of P2P file sharing traffic is higher than the one from Skype [29].

As stated above, the motivation for application detection is often to restrict the usage of a specific application or a group of applications. Consequently, as long as traffic classification is done for this reason, application developers will try to make detection as hard as possible in order to allow a widespread use of their software. This can be compared to the fields of Spam/Anti-Spam or Virus/Anti-Virus communities.

4.3 Classification in Dynamic Port Environments

The first measure of application developers trying to prevent their applications from being detected was the usage of dynamically assigned port numbers. With this approach the quality of traffic classification based on port numbers degrades. Consequently, there has been a lot of research on developing new algorithms to classify traffic.

One idea is the usage of payload-based classification [9, 52, 69]. The challenging task for these classifiers is to identify unique payload signatures. For each application a signature has to be identified in the traffic that is only used by this application. Such signatures can be as simple as that the first few payload bytes of a connection have a fixed signature. For detection of P2P applications the signatures are in general more complex. In 2004 Sen et al. [69] proposed a signature-based classification method to identify the major P2P applications present in the Internet in 2004, providing a detailed analysis of the traffic from Gnutella, e-Donkey, BitTorrent and Kazaa as implemented in 2004. Based on this analysis, signatures have been identified that allow classifying traffic accordingly. It has to be ensured that the signatures are as simple as possible to allow operation on high bandwidth links. For example, for the Gnutella protocol it is checked that the first string following the TCP/IP header is "`GNUTELLA`", "`GET`" or "`HTTP`" and if the first string is "`GET`" or "`HTTP`" there must be a field with a subset of predefined strings, see [69] for details.

Once one has identified the unique payload signature this approach enables very accurate classification. The major problem is that signatures have to be updated with each update of the application, or even when a new application is developed which uses the same signature as an existing application. In this case a new signature for the new application is needed and the signature of the existing application has to be refined.

As applications hiding traffic update their traffic payload on a regular basis, together with the fast increase of networked applications, especially in the P2P field, the time consuming identification of new signatures based on each change became inadequate. A new research field in traffic classification was introduced, where the concept of machine learning is used to detect a range of applications within the traffic [25, 43].

For machine learning classification the requirements presented at the beginning of this

chapter play a central role, as signatures are generated based on an automatic process. Haffner et al. [25] present results from three different machine learning approaches, namely Naive Bayes [64], AdaBoost [66] and Maximum Entropy [4]. The signatures are solely based on packet payload. Machine learning has been performed with pre-classified traffic. Traffic is classified into seven different groups of applications which are FTP control, SMTP, POP3, IMAP, HTTPS, HTTP and SSH. The results are interesting because of two aspects, applications can be detected without any TCP/IP header information and even signatures for encrypted flows can be gathered. Signatures for encrypted flows are based on the plain section of connection setup of HTTPS and SSH. The main weakness of this approach is that it has been only evaluated for applications that mostly use well known port numbers and thus can also be classified using port-based classification [36]. Nevertheless, such an approach together with port-based classification can increase overall classification results.

4.4 Classification when Traffic is Encrypted

As payload-based approaches lead to good results once a signature for an application has been identified, application developers have changed their strategy from changing traffic profiles upon each update to using encrypted payload or partially encrypted payload.

For partially encrypted payload, payload-based signatures may still be based on the plain section of connection setup, which has already been demonstrated for HTTPS and SSH in [25]. For fully encrypted traffic, payload-based signatures fail to identify the traffic appropriately. New algorithms have to be developed that are capable of defining signatures for encrypted traffic.

For encrypted traffic there exist two major research directions:

1. Packet-based classification,

2. Host/social information based classification.

The difference between the two approaches is mainly the question: "What is the central entity used in classification?" For the first approach single flows are used for classification. For the second, it is the behaviour of a host that is used for classification.

4.4.1 Packet-based Classification

For packet-based classification the packets of a single uni- or bi-directional flow are used to decide to which application a flow belongs [13, 20]. Such classification algorithms do not use information that is based on packets from other flows. Encrypted payload packet-based traffic classification algorithms can use the following features for the definition of signatures:

1. header fields,
2. packet size,
3. inter-packet gaps,
4. number of transmitted packets,
5. sequence of packets between hosts.

Packet-based traffic classification uses a subset of the above features. Many traffic classification algorithms, for example [1, 8, 45, 67], use information gathered from header fields. Figure 4.1 shows which header fields are likely to be used. From the IP header mainly source/destination IP and the protocol field are used. Together with source and destination port from the TCP header we have the classical 5-tuple for classification. Recent detection algorithms such as [1] also use the FLAGS of the TCP header for classification, in this special case the PUSH (PSH) flag is used, which is a flag evaluated at the receiver that indicates that this TCP segment should be immediately forwarded to the application and not to wait for further segments before forwarding it. All other header fields are marginally used in traffic classification approaches.

Beside header fields the packet size of one or several packets can be used. For example the size of Packet 1 is 10 bytes, size of Packet 4 is 40 bytes, size of Packet 2 and 3

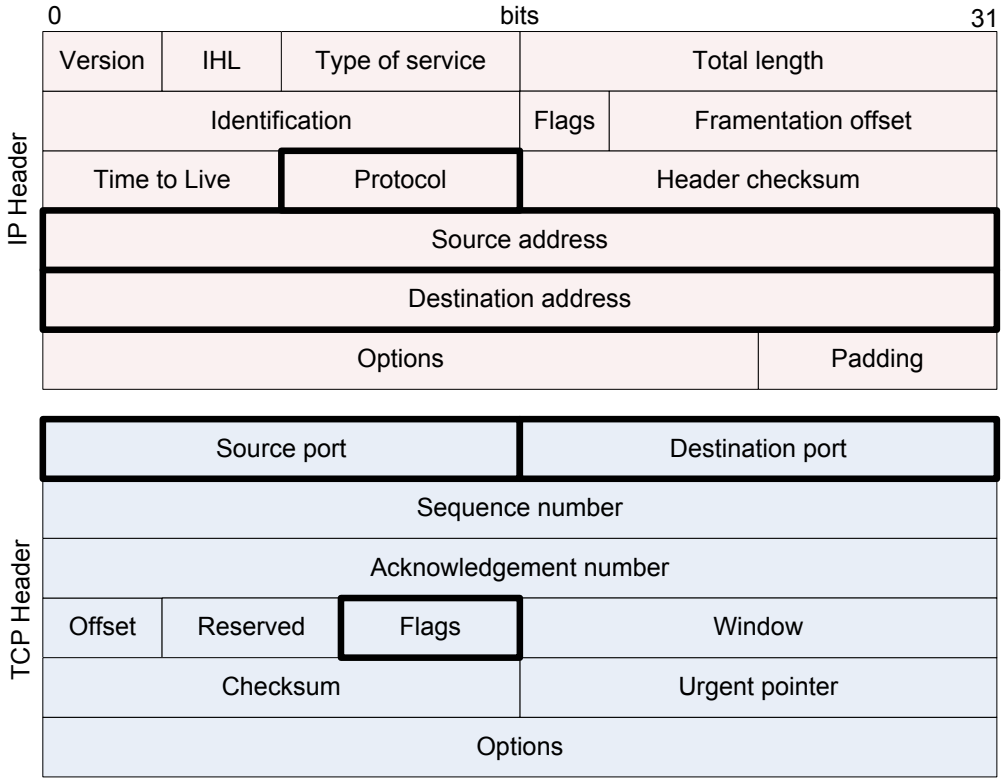


Figure 4.1: TCP / IP Header field usage for traffic classification

are varying. Algorithms that effectively use inter-packet gaps are rare because of the fact that such approaches are prone to changes in the round trip time, as inter-packet gaps strongly depend on the delay between sender/receiver and receiver/sender. Such algorithms have to be customized to specific connections, and, due to the changing delay in networks, the overall quality is still negatively influenced. Counting the number of transmitted packets is mainly used for short flows where only a few packets are transmitted. A sequence of packets means that the signature uses the information that the first packet goes from receiver to sender, the second and third one from the sender to the receiver and so on.

Efficient flow information based classification approaches use a combination of a subset of the above features. In the following, two examples of classification based on information gathered from flows are presented. Both examples are taken from [1], which uses signatures based on information from single flows to classify Skype traffic.

Figure 4.2 shows the signature used to detect a long Skype UDP probe. The signature uses the sequence of packets between the hosts, Probe 1 and 3 in one direction and

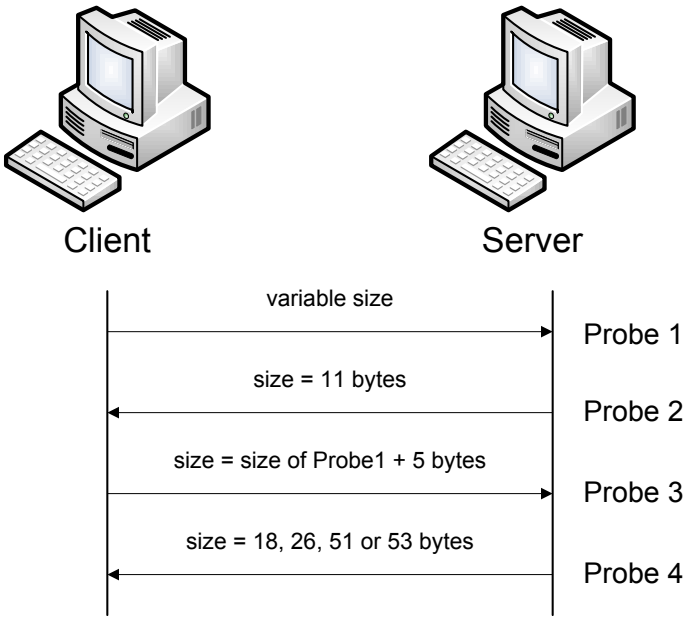


Figure 4.2: Long Skype UDP probe (based on [1])

Probe 2 and 4 in the opposite direction. The number of transmitted packets is four. Further information about the packet size is used in several variants. For Probe 2 the exact size is known, whereas for Probe 3 only the offset compared to Probe 1 is known and for Probe 4 we know that the size is one out of a set.

Figure 4.3 shows the signature of the Skype TCP handshake message. The main difference is that header information is used and that only for two out of six packets information about the size is known. From the header fields the information about the PSH flag is used.

4.4.2 Host/Social Information based Classification

Host/Social information based classification algorithms are using information gathered from several flows to predict the presence of an application on a specific host and classify flows according to host behaviour [28, 31, 35].

Karagiannis et al. [35] present a traffic classification approach that uses host/social information about a host to classify flows from and to this host. The paper proposes three levels of classification. The social level only classifies hosts based on popularity, which is based on the number of hosts communicating with. On the functional level

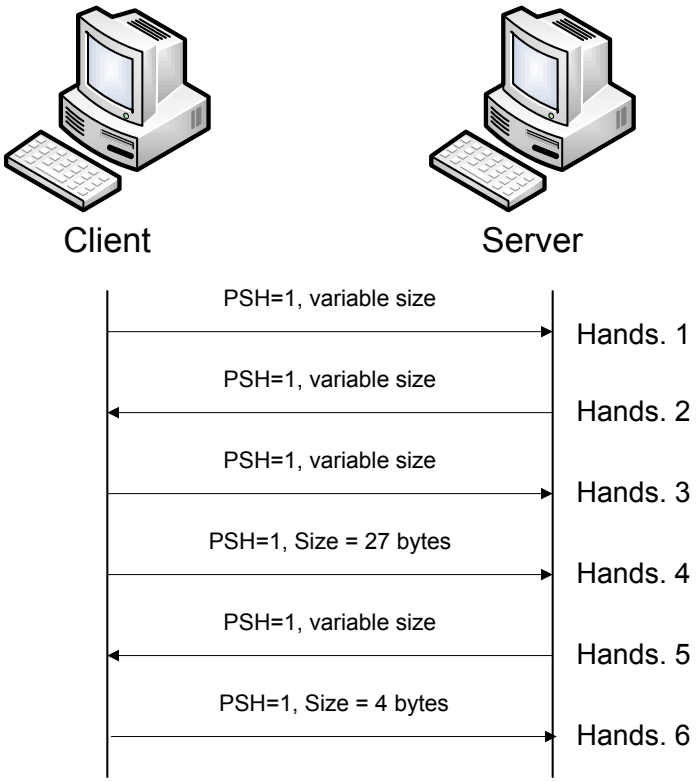


Figure 4.3: Skype TCP handshake (based on [1])

hosts are classified into provider and consumer hosts. Finally, the classification level that is the most relevant one for the work presented here is classification on application level. Classification is based on six heuristics:

1. transport layer protocol,
2. the relative cardinality of destination sets (IPs vs ports),
3. average packet size of the flow,
4. similar behaviour as other hosts (community),
5. detection of a service on a specific host is used for classification of other hosts (recursive),
6. presence of payload.

Host or social based classification is often used in combination with other algorithms. The detection of the Skype TCP handshake message as presented in Section 4.4.1 can

be further enhanced by using host-based information. A precondition for a Skype TCP handshake message is a Skype UDP probe where the port on the server side is the same for the UDP probe and the TCP handshake. Consequently, the classifier can use this host behaviour to increase the overall classification quality.

4.5 Recent Traffic Classification Approaches

Recent research activities show two trends for traffic classification:

1. a detailed and specialized algorithm for detecting a single application,
2. a widespread combination of algorithms for detecting a large set of applications.

Concerning the specialized approach, Skype detection is one field that gained high interest in research. As Skype uses a proprietary protocol where almost all of the traffic is encrypted, the detection of Skype flows is challenging. Skype uses randomly chosen ports, encrypts the payload, or even hides itself behind HTTP or HTTPS connections. Together with the widespread usage of Skype, the detection of Skype traffic is of great interest to the research community. Within the last years a huge number of papers on Skype detection has been published, e.g., [1, 5, 41, 73, 77].

The features used for Skype detection in the aforementioned approaches are widely spread. Header fields, payload-based signatures, statistics on the payload and signatures based on packet directions are as well used as information gathered not directly from the packets of the flow [73].

For example the authors of [77] only use statistics about flows for the detection of Skype hosts. Such an approach allows operating in a privacy preserving environment on high speed links as only marginal information is processed.

The design of traffic classification for a large set of applications is challenging, because a large set of applications needs a large set of signatures for classification. Defining signatures without any machine learning support is almost impossible because of the frequent changes in applications and the introduction of new applications. A major

research direction is to use approaches that generate their signatures by themselves [25, 26, 43, 46]. These algorithms are based on supervised or non-supervised learning. For supervised learning, a training file with pre-classified traffic for all the potential applications is needed which still needs manual classification of a subset in advance.

Most of the proposed algorithms are only applicable to a small subset of network protocols [14, 26]. A novel approach that can potentially be used for a wide range of applications, called statistical protocol identification (SPID), is presented in [26]. The authors use thirty attributes to classify traffic using the Kullback-Leibler divergence [38], which is also known as relative entropy.

The major problem in actual traffic classification is the missing availability of trace files to allow a comparison or evaluation of the algorithms based on a mutual basis. The reasons for this are privacy concerns as well as the problem how to retrieve the ground truth for a trace file. To deal with privacy concerns, one can remove the payload and anonymize the header fields. However, the resulting trace file would be of limited value as many classification algorithms would not be able to operate on such a trace. To retrieve the ground truth, Szabo et al. [74] propose installing a network driver that marks all outgoing packets with an application-specific marking. To change the network driver on all hosts in the network seems to be challenging. Even if the privacy problem and the ground truth problem can be solved, there is still the problem to have a trace file that represents a mixture of the traffic that would be present in reality. In particular, differences in access and backbone networks influence the quality of traffic classification [82].

Currently research is far away from having a traffic classification approach that is applicable on a wide range of applications and a wide range of network links. Simple questions like “How much of my traffic is encrypted?”, “What applications are present on the network?”, “To which extent is P2P traffic used in my network?” or “Which application should be used for traffic classification?” cannot be generally answered accurately.

4.6 Online vs Real-Time Classification

In traffic classification the term real-time is often used in a way that is not in conformance with the actual definition of real-time [17, 80]. A real-time system has to satisfy explicit bounds on the response times. In case of a traffic classification approach, this would mean that there exists an upper bound for the time between receiving the first packet of a new flow and the time when the flow is classified. Many traffic classification algorithms claim themselves to be real-time, but need several packets to classify a flow. As inter-packet times depend on a lot of external factors, such as round trip time or dropped and retransmitted packets, it is not possible to give an upper bound for the time needed to classify the traffic of a flow.

In traffic classification the term "real-time" is not so strictly defined. Algorithms that are capable of classifying traffic on high-speed links are often called real-time classifiers. What is often meant by a "real-time" classifier here is an online or live classifier.

Further algorithms that only need a few packets at the beginning of the connection are called "real-time", as once you have received the specified number of packets an upper bound for the classification time can be given. The major question that arises here is: "Is the approach real-time capable from receiving the first packet of a connection, or is it able to classify a flow in real-time once all the needed information is received?"

Within this work we define real-time such that an upper bound for the response time between receiving the first packet and classification of the flow has to be specified.

4.7 Traffic Classification using Entropy

As more and more network traffic is encrypted, entropy-based classification algorithms have gained interest within the last years. Entropy-based approaches are often used to detect malicious traffic [42, 55, 78].

Wagner and Platter [78] use entropy of traffic parameters such as IP addresses to detect fast changes in network behaviour. For example, a worm can be detected when a large number of hosts start to contact a large number of other hosts where all of them are

using the same destination port, which has a vulnerability.

Lyda and Hamrock [42] use an entropy-based approach called bintropy that is able to quickly identify encrypted or packed malware. The entropy is used to identify statistical variation of bytes seen on the network.

Pescape [60] uses entropy to reduce the amount of data that has to be processed by traffic classification tools. Entropy is used as input for an advanced sampling approach that ensures that sensible information needed to get an appropriate model of the network traffic is still present. Packets not needed for an appropriate model are dropped.

An entropy-based approach which inspired the present work is presented in [55]. The N-truncated entropy for different encrypted protocols is determined. For example, for an HTTPS connection the byte entropy after 256 bytes of payload should be between six and seven. If the value for a specific connection is below this range, it is assumed that the connection is subverted.

5

User Privacy in Computer Networks

During the last years the legislation has enacted a few laws which extend the right for privacy to the world of computer networks [23]. Due to these changes two areas in network operation have to change their operation to be in compliance with these new laws.

Network researchers need traces with network traffic for development and evaluation of new algorithms. Network providers need to monitor their network to ensure proper operation of the network.

Based on the above mentioned laws it is getting more and more difficult for researchers and network operators to fulfil the needed tasks and comply with the law. The most important regulations that influence and motivate our work are:

1. IP addresses are privacy sensitive information even if dynamically allocated,
2. the user has to be informed in advance for which purpose which monitored information is used,
3. only the information that is needed for the specific task should be used.

Based on these regulations, two major parts of a packet have to be handled with care in a privacy preserving environment. On the one hand the IP addresses, and on the

other hand the payload of the packet. Different strategies are applied to them, whereas for IP addresses a potential solution can be anonymization, the payload is often simply stripped off.

Simply stripping the payload would end up with broken applications that need parts of the payload, such as intrusion detection systems (IDSs). An example for such an IDS is an element of a firewall that inspects the payload of packets to detect email worms. Also anonymization cannot neglect the needs of a specific application. Anonymization is often used for IP addresses, where an IP address is replaced by another IP address. A simple approach would be to use a random value for each packet, but if the application that operates on the anonymized data needs to identify which packets belong to which connection a specific IP address has always to be replaced by the prior used IP address. Consequently, there has to be a correlation between the original IP address and the anonymized IP address. The kind of correlation strictly depends on the intended usage of the data. The more correlation is needed, the weaker the anonymization is and thus it may be broken by an attacker that gets the anonymized data.

As the RT-ETD can be used to support operation of applications in a privacy preserving environment, we will now present a framework for privacy preserving secure network monitoring (PRISM).

5.1 Privacy Preserving Network Monitoring

The EU funded project PRISM [75] provides a framework for privacy preserving secure network monitoring. The core concept of the project is that each given purpose can be satisfied in a privacy preserving environment. An example for such a purpose can be to detect all Skype flows. Consequently, the central entities are no longer monitoring applications but are purposes, where a specific monitoring application can be used to satisfy these. The usage of the purpose as central entity is mainly motivated by two aspects. Firstly, an employee of an operator wants that a purpose is satisfied and not that a monitoring application is working properly even if thinking in an application focused manner, as we are used to it. For example, someone would like to have the

TCP Statistic and Analysis Tool (TSTAT) from the University of Turin running in a privacy preserving environment to detect Skype traffic.

Secondly, applications often satisfy a wide range of different purposes. In a privacy preserving environment it has to be clearly stated to the network user what the monitored traffic is used for. This information cannot be given to the user for such applications as the usage is not limited to a specific purpose.

As TSTAT is a powerful tool and has a wide range of operations, a full functional TSTAT would not be able to operate in a privacy preserving environment. What is possible is to have a privacy preserving Skype traffic detection by using TSTAT. So the PRISM system satisfies a customer purpose by using a specific monitoring application for this. To ensure privacy preserving monitoring the PRISM system provides a two-tier approach which will be presented in the next section.

5.1.1 Architecture

The PRISM architecture, shown in Figure 5.1, targets a line speed of 1 Gbit/s for privacy preserving network monitoring. A monitoring application is split into 3 main parts:

1. front-end for short time, low processing effort on-the-fly analysis,
2. back-end for long time analysis and storage,
3. remaining part of monitoring application that stays outside the PRISM system.

The front-end mainly provides on the fly operations that need no or only little buffering of information. It consists of 4 major processing steps where for different purposes the chain looks different. The first step is copying the traffic from the wire for further processing. Following to this is the major step in the front-end and can have various subtasks.

The first analysis step is a filter based on packet header information, where the length of each individual packet can be cut. It is cut to the length that is needed in further

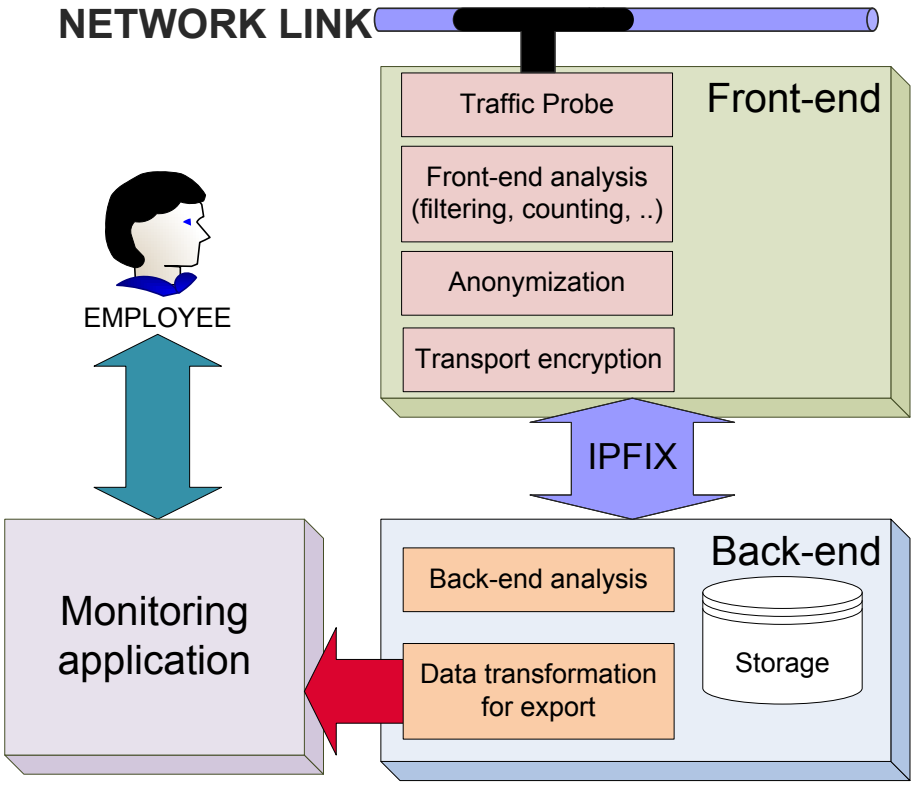


Figure 5.1: PRISM architecture

analysis steps. After this the chain may be completely different for each purpose. In this step the traffic is further reduced, results are calculated or complex filtering approaches are applied.

Privacy sensitive information which is not needed in back-end analysis functions is already anonymized in the front-end. The last step before the traffic is transmitted is to encrypt the traffic to be protected from being tapped on the wire. Traffic between front-end and back-end is transported using the IPFIX (Internet Protocol Flow Information Export) protocol, which is a standardized protocol for exchanging monitoring information.

The back-end decrypts the traffic and performs long term analysis. Such analysis can for example be exponentially weighted moving average or payload inspection over a reasonable amount of packets. Analysis results are then stored in an encrypted data base for later usage. The data transformation for export operation transforms the data into the format which the external monitoring application requires.

The external part of the monitoring application can be a legacy application that only operates on a subset of the traffic if the PRISM system is capable of filtering the traffic in that way that user privacy is not violated. Or, if all processing steps include privacy sensitive information, the external part of the monitoring application will get the final results for presentation purposes only.

The whole PRISM system is surrounded by a privacy preserving access control system, which ensures that only dedicated users with a dedicated role are allowed to launch a privacy preserving monitoring task. The access control system is based on an ontology.

5.1.2 Development of a Privacy Preserving Application

For the design of a privacy preserving application, the first question is: "What purpose should be satisfied?" There has to be a closer look on what information has to be processed for this purpose. The processed information has to be split into privacy sensitive and non privacy sensitive information, where processing of the latter can be done outside of the PRISM system. Processing of privacy sensitive information has to be performed in the PRISM system.

Within the PRISM system the processing is split into on the fly tasks performed in the front-end and long term analysis in the back-end. Based on the monitoring purpose there exists a wide range of possibilities for privacy preserving monitoring. When privacy sensitive information is processed, it may be needed to split the application into 3 parts where the external part is only for visualization. The other extreme would be that the application can be fully operated outside of PRISM if privacy sensitive information can be removed by filtering. For most applications, the solution is somewhere in between.

5.2 Content Providers and Privacy

The RT-EDT supports network operators in privacy preserving network monitoring. The second major area of privacy threats in computer networks are content providers, where the RT-EDT is not able to support user privacy. Content providers are often

collecting as much information as possible about their customers. The main reason for this is business, as the more information about a customer they have, the more likely it is to make revenue out of this knowledge.

Content providers are often hiding the collection of data behind laws of nations without any user privacy legislation. Collecting privacy sensitive information is widespread and starts with simply logging each user's behaviour on the website to allow sending personalized offers per email.

An approach where user privacy is simply ignored is when content providers start to collect as much data of a user as possible, by using all methods that are available. Such approaches include phishing of account details of other websites.

A recent paper [79] shows an approach where, based on the browser history of a customer, the content provider is able to identify which websites have been visited and thus to identify which social networks the customer is using. This even includes the ability to identify facebook groups the customer joined. As a user, one should be aware of the fact that, when all friends are interlinked within a social network webpage as facebook, facebook is able to see into one's social life. What users are often not aware of is that a content provider of a different webpage can do this as well.

Content providers should change their behaviour, to obfuscate the social network of their users, and make it more difficult to identify individual users. For such obfuscation entropy can be used as a kind of measure to keep the level of obfuscation high, and reduce the risk of being infiltrated. The usage of entropy in this domain is left open for future work.

6

Real-Time Detection

This chapter presents a novel approach for traffic classification based on information solely gathered from content of the first packet of a flow. The first packet in this context is defined as the first packet sent by a UDP connection and for TCP the first packet is the one that follows the three-way handshake. The term flow is defined as bi-directional flow based on the 5-tuple.

The goal of the proposed approach is to operate as a real-time encrypted traffic detector (RT-ETD). The algorithm is targeted at being used as a traffic filter in a privacy preserving environment, where the task is to filter traffic and forward only flows which are encrypted (i.e. remove flows with privacy sensitive payload).

The proposed system is based on several modules that can be combined to meet the requirements of various purposes. Customizing of the algorithm will be dealt with in Section 6.6 where the proposed algorithm is used as a privacy preserving real-time filter for Skype detection, with the Adami Skype detector [1]. Furthermore, the approach is customized for SPID [26] which is a classifier that detects multiple encrypted protocols.

The proposed algorithm can be divided into five different modules that are used for classification of traffic.

1. Port-based classification,
2. Entropy-based classification,
3. Coding-based classification,

4. IP(-address)-based classification,

5. Content-based classification.

Each of them will now be explained in more detail.

6.1 Port-based Classification

Within the range of well known ports [27] a few ports exist that are solely targeted at applications that use encrypted communication. The most important ones are shown in Table 6.1. The proposed algorithm offers the possibility to use a port-based classifier which is based on the assumption that traffic on these ports is encrypted and consequently can be forwarded without impacting privacy.

We assume that there is no application that transports unencrypted traffic on these ports. If a user or application developer does this, the reason is most properly that this traffic should be hidden, to circumvent rules or regulations set by an administrator. We assume that forwarding such traffic is not a violation of their privacy as they have to be aware of the fact that they are using well known encrypted ports for unencrypted traffic.

Port numbers greater than 1023 are not taken into account for the port-based classifier as these ports may not be so strictly bound to a specific type of application.

The main motivation why we are not using the entropy based algorithm here is the fact that port-based classification is faster, and that on these ports the first packet is often an unencrypted initialization message. Therefore a solely entropy-based classification approach would always drop that traffic.

For example, an HTTPS connection starts with a `Client Hello` message, this message has a partially fixed content and does not have random payload. Still, the communication is encrypted within an HTTPS connection.

Name	Port	Application
SSH	22	The Secure Shell (SSH) Protocol
NSIIOPS	261	IIOP Name Service over TLS/SSL
HTTPS	443	HTTP protocol over TLS/SSL
DDM-SSL	448	DDM-Remote DB Access Using Secure Sockets
NNTPS	563	NNTP protocol over TLS/SSL
sshell	614	SSLshell
LDAPS	636	LDAP protocol over TLS/SSL
CORBA-IIOP-SSL	684	CORBA IIOP SSL
IEEE-MMS-SSL	695	IEEE-MMS-SSL
FTPS-data	989	FTP protocol, data, over TLS/SSL
FTPS	990	FTP protocol, control, over TLS/SSL
telnets	992	telnet protocol over TLS/SSL
IMAPS	993	IMAP4 protocol over TLS/SSL
IRCS	994	IRC protocol over TLS/SSL
POP3s	995	POP3 protocol over TLS/SSL

Table 6.1: Encrypted well-known port numbers

6.2 Entropy-based Classification

This section presents the core algorithm of the RT-ETD. We use the approach of Olivain and Goubault-Larrecq [55], which is described in Chapter 2, to determine whether the payload is encrypted or not. While Olivain and Goubault-Larrecq evaluate the payload of several packets, within this approach only information gathered from the first packet of the flow is used. The core concept is to estimate the N-truncated entropy of the payload of the first packet and compare this value to the N-truncated entropy of uniformly distributed payload. Based on the difference between these two values, it is decided whether the flow is encrypted or not.

In Chapter 2 we present an approximation of $H_N(U)$, as an alternative approach Olivain and Goubault-Larrecq suggest using standard Monte-Carlo method where a large number of words w of length N are drawn at random and the average is taken as $\hat{H}_N(U)$. Also for an estimation of the confidence intervals the standard Monte-Carlo method is suggested to be used. Within this work the standard Monte-Carlo method for $\hat{H}_N(U)$ and for an estimate of the confidence intervals is used. For each N from 1 to 1472 100.000 words were generated, using the Matlab [76] function `rand`. $\hat{H}_N(U)$ and the confidence intervals are based on this experiment.

What is not evaluated in [55] is the influence of choosing the byte entropy which means

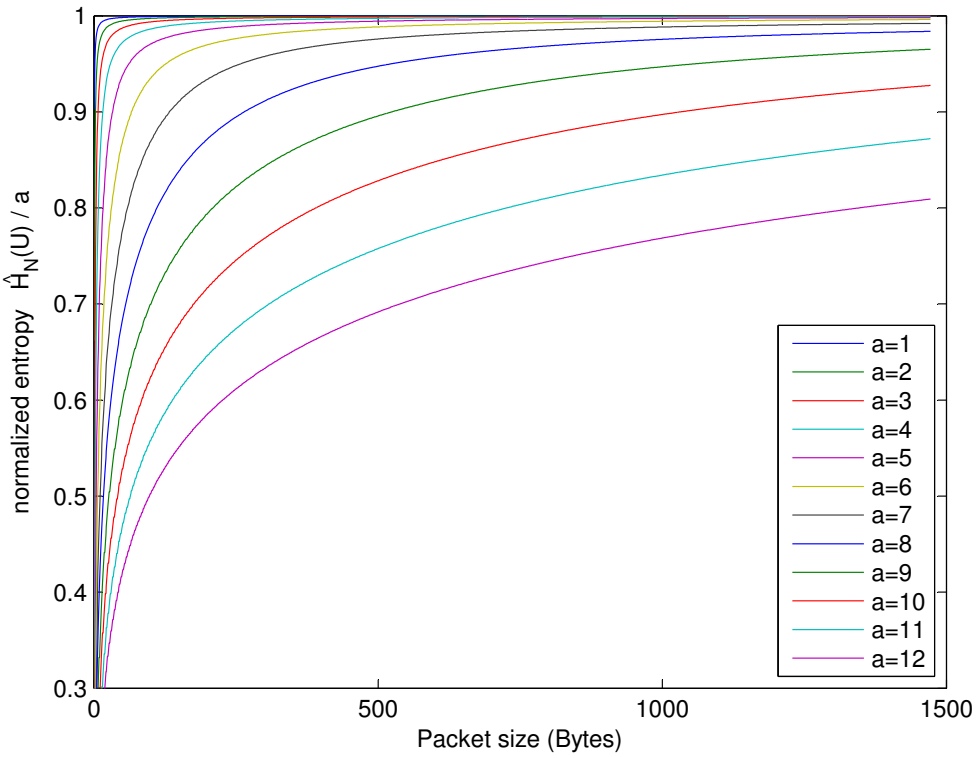
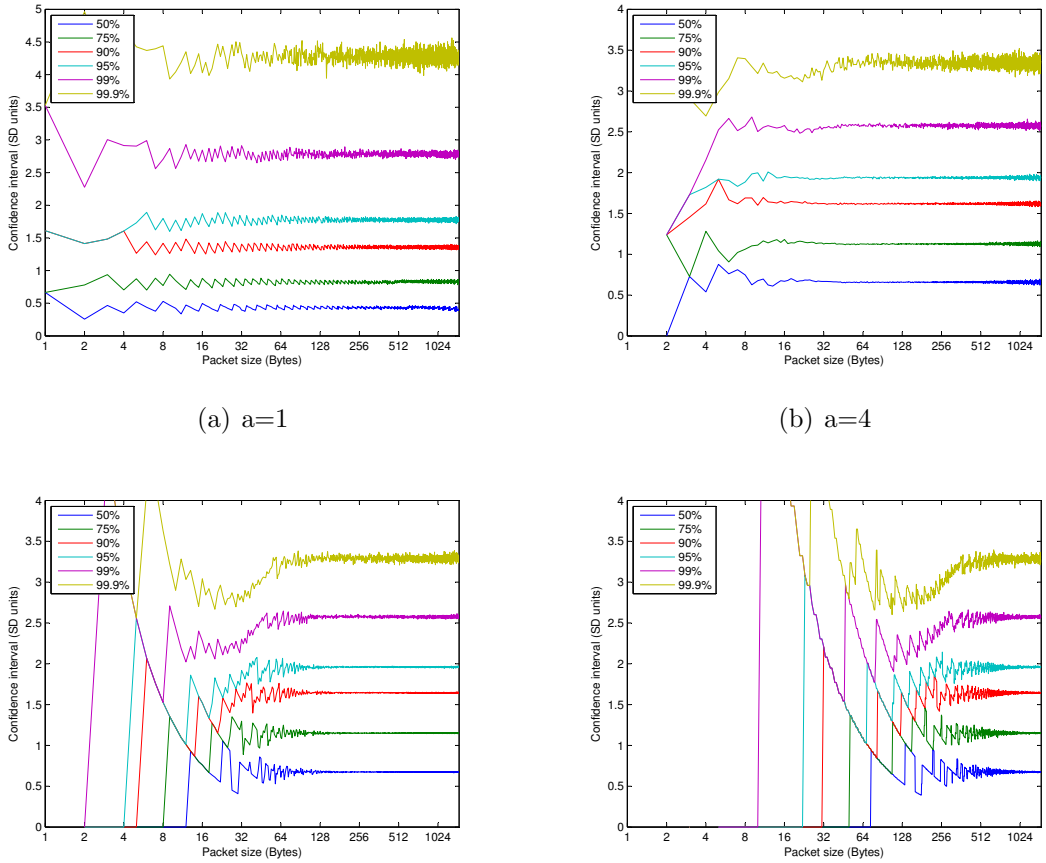


Figure 6.1: Influence of a on entropy for given word length based on Monte-Carlo

"m=256" without taking into account for example, bit entropy or nibble entropy. We have performed a study on the influence of the size of the alphabet m. We define a to be the number of bits used for the alphabet which means $m = 2^a$. For a we are using 1 to 12 bit and compare the results for entropy and confidence intervals. The entropy values are normalized by dividing by the number of used bits, such that all values are between 0 and 1.

Figure 6.1 shows the influence of choosing a for given words of length N based on Monte-Carlo. The major aspect is that the larger we choose a the smaller $\hat{H}_N(U)/a$ will be. All curves tend to one. As in an undersampled environment the MLE estimator underestimates the entropy it is obvious that for larger a longer words will be needed to get an estimate equal to one.

As basis for an estimation of the confidence intervals we use the same Monte-Carlo method as for $\hat{H}_N(U)$. The results of the Monte-Carlo method are fed into the Matlab [76] function `prctile`, which calculates the borders of the confidence intervals. We



(a) a=1

(b) a=4

(c) a=8

(d) a=12

Figure 6.2: Normalized length of the confidence intervals

then calculate the length of the confidence intervals by subtracting the lower border from the upper border.

Figure 6.2 plots the length of the confidence intervals for $a = 1$, $a = 4$, $a = 8$ and $a = 12$. The length of the confidence intervals is normalized by the standard deviation (SD) to allow easy comparison and divided by two to show them in $\pm SD$ units. The plots for the remaining eight a look similar. Based on Figure 6.2 it is obvious that choosing a confidence bound of plus/minus three times the standard deviation will lead to at least a 99% confidence bound independent of a. Consequently we are 99% sure that entropy of uniform distributed payload is within this range. Only for short payloads this is not true. An explicit look on short payloads will be given later. Values between $H_N(U) + 3 \times SD$ and H are very likely to be encrypted, but we stay to the confidence bound of $3 \times SD$ as based on Figure 6.2 it can be seen that the 99.5%

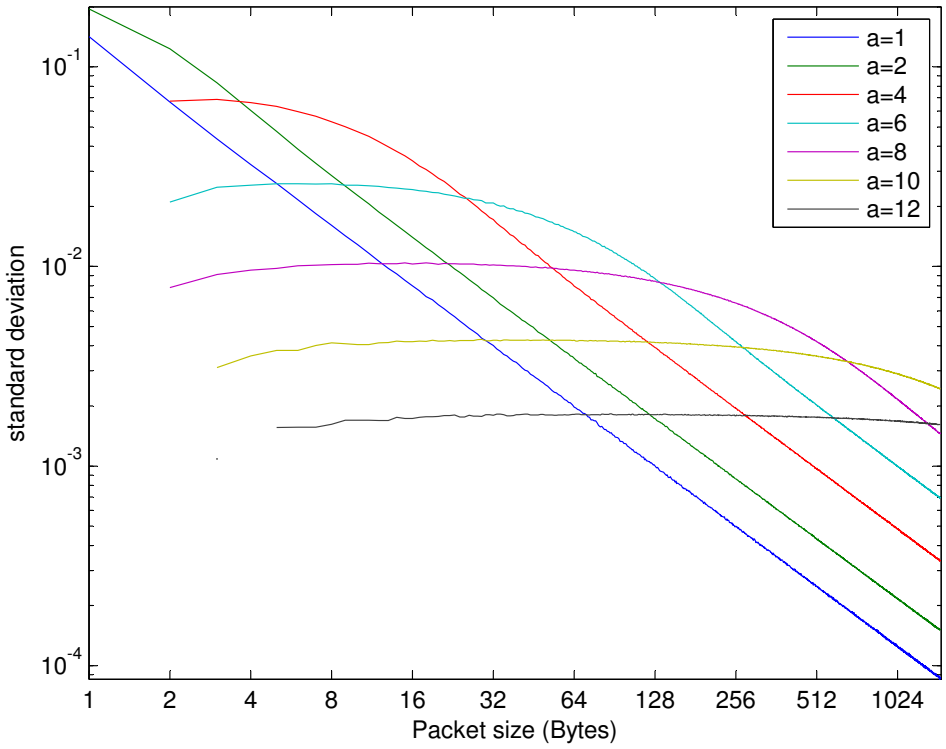
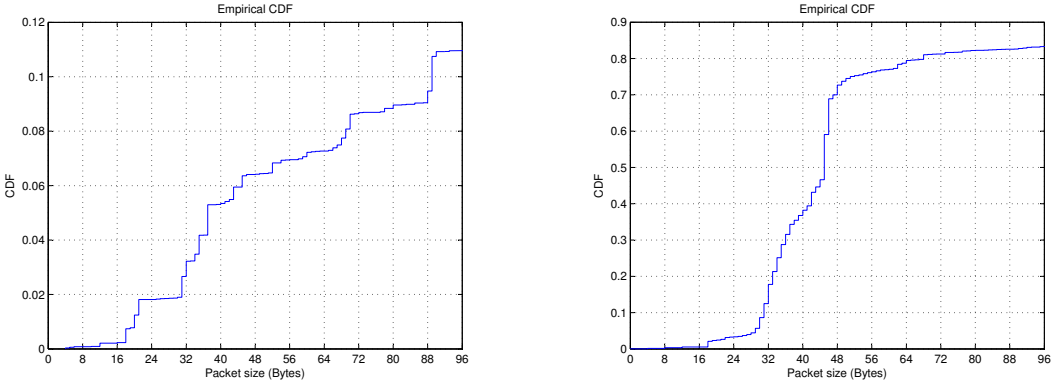


Figure 6.3: Standard deviation

percentile is within the range of $H_N(U) + 3 \times SD$, consequently at most 0.5% are beyond $H_N(U) + 3 \times SD$. The evaluation in Chapter 7 will show that the differences are marginal.

As for the confidence bound we are using three times the standard deviation, a detailed look on the standard deviation for different a is given next. Figure 6.3 shows the standard deviation for different a. For packet size up to 71 bytes $a = 12$ has the smallest standard deviation whereas for packet size greater than 71 bytes $a = 1$. But if we choose $a = 12$ for small packet sizes the 99% confidence bound will be larger than three times the standard deviation. As argued in [59] for a test for uniformity the word should at least have the length of $\sqrt{m}$, which would in the case of $a = 12$ be 64 words of length 12 (or equivalent 96 bytes). The reason for this is that, with shorter words there is at least one distribution that cannot be distinguished from the uniform distribution, for details see [59]. The consequence of this would in our case be that unencrypted traffic is forwarded.



(a) CDF TCP (b) CDF UDP

Figure 6.4: CDF for UDP/TCP packets

The confidence bound of $3 \times SD$ ensures that we are 99% sure that entropy of uniform distributed payload is within the range of $3 \times SD$, which is only one goal that should be retrieved, further unencrypted traffic should be removed. As the standard deviation is small we are confident that only a small fraction of unencrypted traffic will be in this range. The evaluation in Chapter 7 shows that in practice the decision bound works well. Further theoretical consideration is left open for future work.

To determine whether the RT-ETD has to be operational below 96 bytes let us have a look at a real world network trace. We have collected 2GB of traffic in a local cable network provider's network. Figure 6.4 plots the CDFs (cumulative distribution function) of the packet size of the first packet of UDP/TCP flows. It is obvious that in case of UDP flows the algorithm has to be operational below 96 bytes, as more than 80% of the first packets are shorter.

According to Figure 6.4 it is obvious that our approach has to be operational for a packet sizes of eight or at least sixteen bytes. For a packet size of sixteen bytes, about 0.5% of the UDP flows cannot be evaluated. We assume that this low fraction of flows can be neglected.

Consequently we need to use an a that is smaller than 9 as, according to [59], greater values need longer words than 16 bytes.

To choose $a = 1$ may be not the optimal choice for a completely different reason. In this case it is only determined whether the number of zeros and ones is equal within

the word or not. For example, the hexadecimal (HEX) sequence "`0F0F0F0F0F0F`" has the maximum entropy of 1 although it is clearly not random.

To find the optimal a is not trivial. We are going to investigate this in more detail by evaluating the influence of a on the filtering performance in Chapter 7.

6.3 Coding-based Classification

The coding-based classifier utilizes the fact that many protocols use ASCII encoded plain text. For example the payload of the first packet of a POP3 message can look like "`+OK Dovecot ready...`"[1] which is equal to "`2b 4f 4b 20 44 6f 76 65 63 6f 74 20 72 65 61 64 79 2e 0d 0a`" in HEX. Based on the HEX values the above plain POP3 payload looks random. For this example, based on $a = 8$, $\hat{H}_N(U)$ is 4.25, the entropy based on MLE is 4.02. The standard deviation is 0.082, as 4.02 is within the range of $4.25 - 3 \times 0.08$. Consequently, the entropy-based classifier would assume that the message is encrypted. The first packet partially consists of a free configurable text, in this example "`Dovecot ready.`". A different server is instead using "`Hello there.`". Depending on this free configurable text, the entropy-based classifier decides whether the flow is encrypted or not. For the packet with the payload "`+OK Hello there...`" $\hat{H}_N(U)$ is 4.10 and the MLE entropy is 3.68, which is outside of $4.10 - 3 \times 0.083$, consequently this flow is assumed to be plain. For a human being it is clear that both messages are unencrypted. It is obvious that there is a big text section in the payload.

A shown by the example above our entropy-based classification for encrypted traffic has a too high rate of false positives. To overcome this shortcoming of the entropy-based approach we add a further check. We assume that for plain text messages the payload is encoded in ASCII or ANSI where coding values 32 to 127 are used for printable characters.

For our entropy estimation the entropy of the example above is within the given confidence bound. Nevertheless it is obvious that it is very unlikely that a random source

[1] The two dots at the end are non printable characters

would produce such a word. The probability that a character from a random source will be in the range from 32 to 127 is about 37.5%. Especially if at the beginning of the packet a large fraction of characters is in this range the payload is most likely unencrypted. Consequently we added a check that if the fraction of bytes with values between 32 and 127 is greater than 75% we assume that the flow is unencrypted. 75% means that the fraction of bytes in the range from 32 to 127 is twice as high as that of a random source.

As we want to reduce the processing time we do not evaluate the full payload of the packet. We only evaluate the first few bytes. Only taking into account too few bytes would lead to a high rate of misinterpretations. We exemplarily choose to use a maximum of 96 bytes, which is the length of the range from 32 to 127. An evaluation for different values for this is left for future work.

A more complex possibility would be to check for n-grams in the payload. For example we can count the number of occurrences of sequences in the range from 32 to 127 of length 3,4,5 and so on. With this aspect we would not only take into account the fraction of letters and numbers but also the grouping of those. This approach would allow a more detailed analysis of the content. As the simple approach only taken into account the fraction, in the range from 32 to 127, works fine for our purpose we do not use n-grams for now.

An analysis based on real network traces has shown that the coding-based classifier does not drop any UDP flows, consequently the approach is only used for TCP flows.

6.4 IP-address-based Classification

What has not been taken into account until now is the knowledge that a specific IP address or a range of IP addresses only uses encrypted communication or is only offering information that is not privacy sensitive. The IP-address-based classification module allows specifying IP addresses that should be forwarded without further checking for port numbers, entropy or coding.

Until now we have been concentrating on identification of encrypted traffic. Traffic

classification often has the goal to classify all the traffic from a specific application. Often applications that are using encrypted traffic do not encrypt a small fraction of traffic. An example of such an application is Skype where the HTTP Update is unencrypted. Skype uses a single host for the updates, consequently the IP-address-based classification module can be used to forward Skype HTTP Update messages. This specific IP does not offer any other service, and the Skype HTTP Update messages do not transport privacy sensitive information, consequently forwarding them is inline with our vision.

6.5 Content-based Classification

This classification module uses very detailed information about the payload of the first packet of an application.

For example the Skype Login starts with a packet that is only 5 bytes long, consequently our entropy-based approach fails. However, based on [1] we know that the first Skype Login packet has a payload of HEX "`22 03 01 00 00`" and the PSH flag set. If our classification algorithm should be used as pre-filter for Skype traffic, we add a classification module that uses this specific knowledge to identify a Skype Login flow. Such an approach can be used for any application where the other modules fail but enough detailed information about the first packet is known.

In general specific knowledge about the first packet can be used to further enhance the filter. Such knowledge can for example be areas of unencrypted traffic that are not taken into account for the entropy classifier or partially fixed values within the payload.

6.6 Full Classification Chain

The classification modules described above are combined to form a classification chain. The decision whether to forward a flow is still based on the first packet, which traverses the blocks shown in Figure 6.5 until one block decides to forward it or it reaches the end of the chain. If the packet is forwarded, the corresponding 5-tuple is added to the

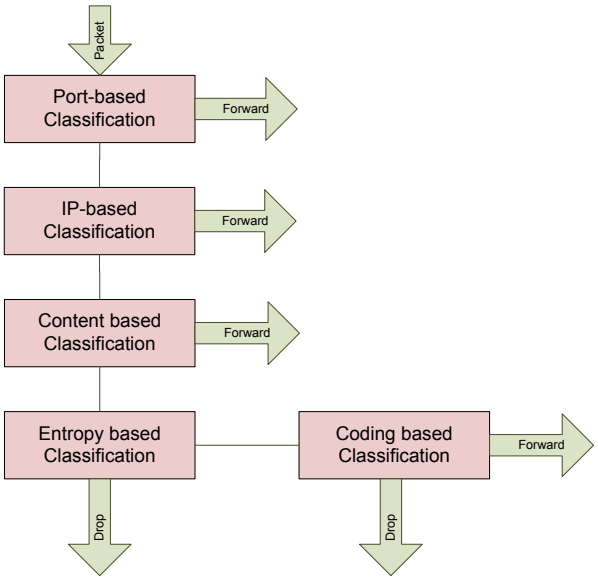


Figure 6.5: Block diagram for classification

list of encrypted flows, otherwise it will be dropped and the 5-tuple will be added to the list of unencrypted flows. After receiving a TCP segment where the FIN flag is set, which is set in the last packet of a connection indicating that all data has been transmitted, the 5-tuple will be deleted from the lists. As there is no such signalling for UDP flows we use a timeout for UDP flows. If for the duration of the timeout we do not receive a packet for a specific 5-tuple it is deleted from the lists.

A major goal in traffic classification is to identify all the traffic from a specific application. Unfortunately, a lot of applications use encrypted as well as unencrypted traffic. Thus our approach removes a fraction of the application's traffic and possibly degrades the detection quality. Therefore we provide the possibility to customize our algorithm to forward application specific traffic that can be identified based on the first packet and is unencrypted. We provide this exemplarily for Skype detection. Furthermore we provide a customization as pre-filter for SPID [26].

6.6.1 Skype Customized Classification

This section presents a filter chain for Skype classification with the Adami Skype detector [1]. Skype traffic can be divided into 4 groups that will be detected by using different modules from above.

1. Flows where the first packet is longer than $\sqrt{m}$ bytes and the entropy evaluation suggest that it is encrypted,

2. Flows on port 443 (based on the first packet it is not possible to distinguish Skype flows on port 443 from normal HTTPS flows. As either is encrypted there are no privacy concerns when forwarding them.)

3. HTTP flows to `ui.skype.com`[2],

4. Flows where first packet is only 5 bytes long and has a fixed payload (see example in Section 6.5).

For the first category of traffic our entropy-based classifier together with the coding-based classifier will be used. Traffic on port 443 will be detected by the port based classifier whereas the Skype HTTP update will be detected by the IP address used for the update which is `204.9.163.158`. The Skype Login flows are detected as presented in Section 6.5. The flow chart for pre-filtering of Skype traffic is shown in Figure 6.6.

We use a few lists to store information. The first one is a list of known Skype IP addresses, we are only using the IP address of the update server `ui.skype.com` (`204.9.163.158`) here. The SYNList stores all 5-tuples where we have received a packet with the SYN flag set, which indicates a TCP connection establishment. A 5-tuple is removed from the SYNList after receiving the first packet containing payload or receiving a packet where the FIN flag is set. Furthermore the 5-tuple will be removed from the SYNList if, 60s after the packet with the SYN flag set, there has not been any data packet. This should prevent the SYNList from growing due to SYN flooding attacks.

A TCP flow which was detected as encrypted will be stored in the ENCRList. The 5-tuple will be removed upon receiving a packet where the FIN flag is set. Encrypted UDP flows are stored in the ENCRList, whereas unencrypted UDP flows are stored in the UNENCRList. The UDP 5-tuples are removed from the lists if there has not been a packet for this 5-tuple for 300s.

[2]This server is solely used for updates of the Skype client via HTTP

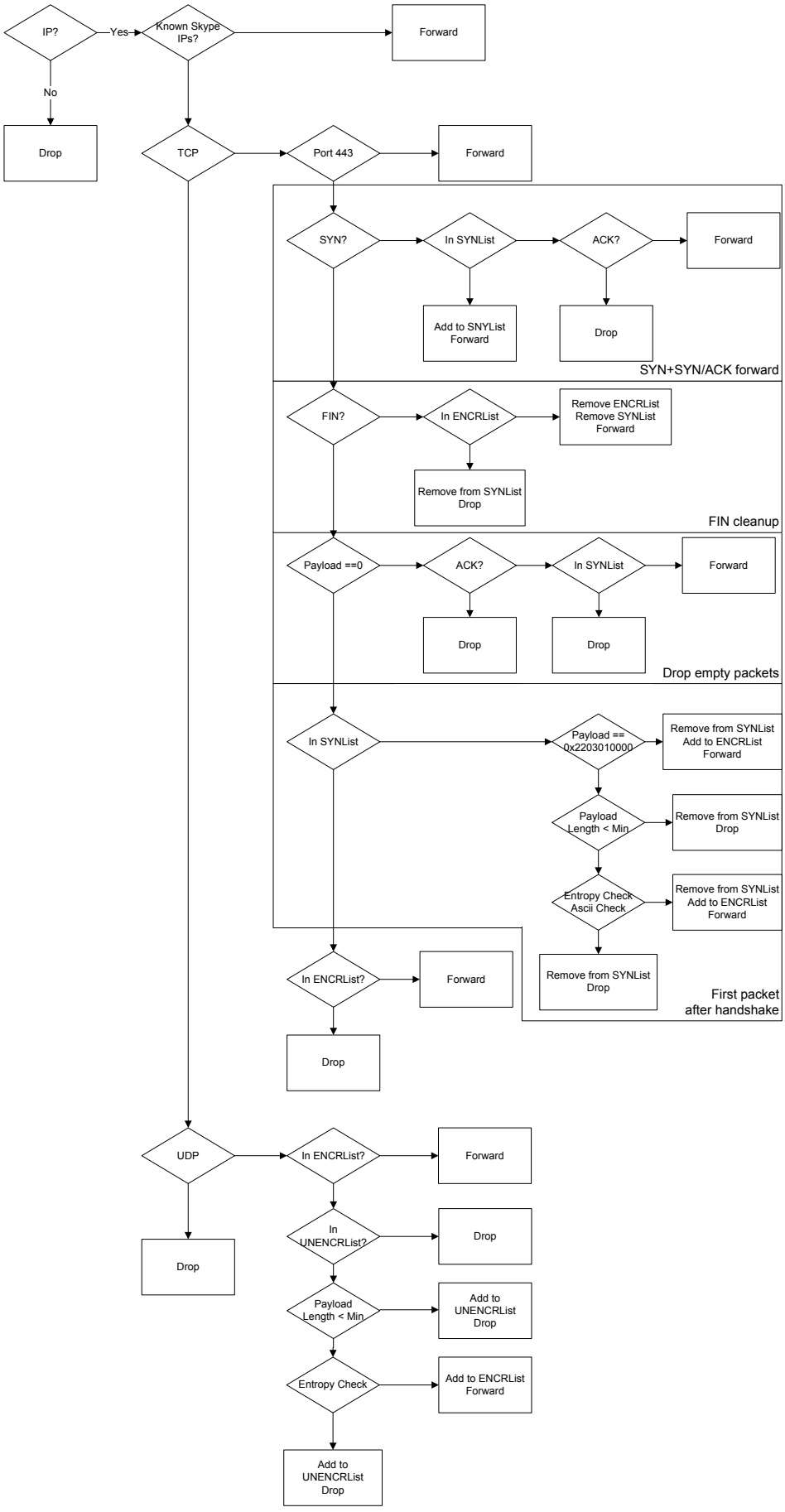


Figure 6.6: Skype pre-filtering flow chart

All horizontal arrows in the flow chart indicate that the answer at the decision point was yes, vertical arrows indicate a no.

First of all non IP traffic is dropped. The next check ensures that all traffic to `ui.skype.com` is forwarded. Next we distinguish between TCP and UDP. For TCP we used a port-based classifier to forward all traffic on port 443. The "SYN+SYN/ACK forward" block ensures that the first two messages of the three-way handshake are forwarded, and that the 5-tuple is stored in the SYNList. The "FIN cleanup" block ensures that when a packet is received where the FIN flag is set the 5-tuple is removed from SYNList and ENCRList. As the Skype detector, for which we perform the pre-filtering, does, after the initial three-way handshake, not evaluate packets with empty payload we only forward empty packets where the ACK flag is set and which are in the SYNList. All other packets with empty payload are dropped. This behaviour is ensured by the "Drop empty packets" block.

The core concept of the algorithm for TCP flows is in the "First packet after handshake" block. It evaluates whether the flow should be forwarded or not, for any decision the flow will be removed from the SYNList. If the flow should be forwarded the packet will be forwarded and the flow will be added to the ENCRList. The "In SYNList" check ensures that this is the first data packet after the three-way handshake. The next check is whether the payload is equal to HEX "`2203010000`" to ensure to forward Skype Login flows. The following check drops all flows where the first packet is shorter than $\sqrt{m}$ as the entropy-based classifier does not work reliably on such short packets. The last check in this block is based on the results of the entropy and the coding-based classifier.

The last check for TCP forward packets of flows in the ENCRList and drops all others.

For UDP the sequence is shorter as there is no three-way handshake to be taken into account. If the flow is already in the ENCRList it will be forwarded, if is in the UNENCRList it will be dropped. If the packet does not belong to a flow in any list, it needs to be evaluated. Packets shorter than $\sqrt{m}$ will be dropped and added to the UNENCRList. Longer packets will be checked with the entropy-based classifier. If decided that the packet is unencrypted it is dropped and the flow is added

to the UNENCRList, otherwise the packet is forwarded and the flow is added to the ENCRList.

6.6.2 SPID Customized Classification

In this subsection we present the usage of the RT-ETD as pre-filter for encrypted traffic that should be classified by SPID [26]. SPID is able to identify several unencrypted applications (DNS, HTTP, POP, FTP, SMTP, ...) and the following encrypted applications:

1. Skype TCP,
2. Skype UDP,
3. MSE,
4. eDonkey TCP,
5. eDonkey UDP,
6. Spotify P2P,
7. Spotify Server,
8. SSH,
9. SSL.

There are two major reasons why our approach can be used as pre-filter for SPID:

1. only processing encrypted traffic (privacy),
2. only processing a fraction of traffic thus making SPID faster.

As SPID uses metrics based on statistical information, a third possibility would be to integrate the entropy-based approach as a further metric directly into SPID.

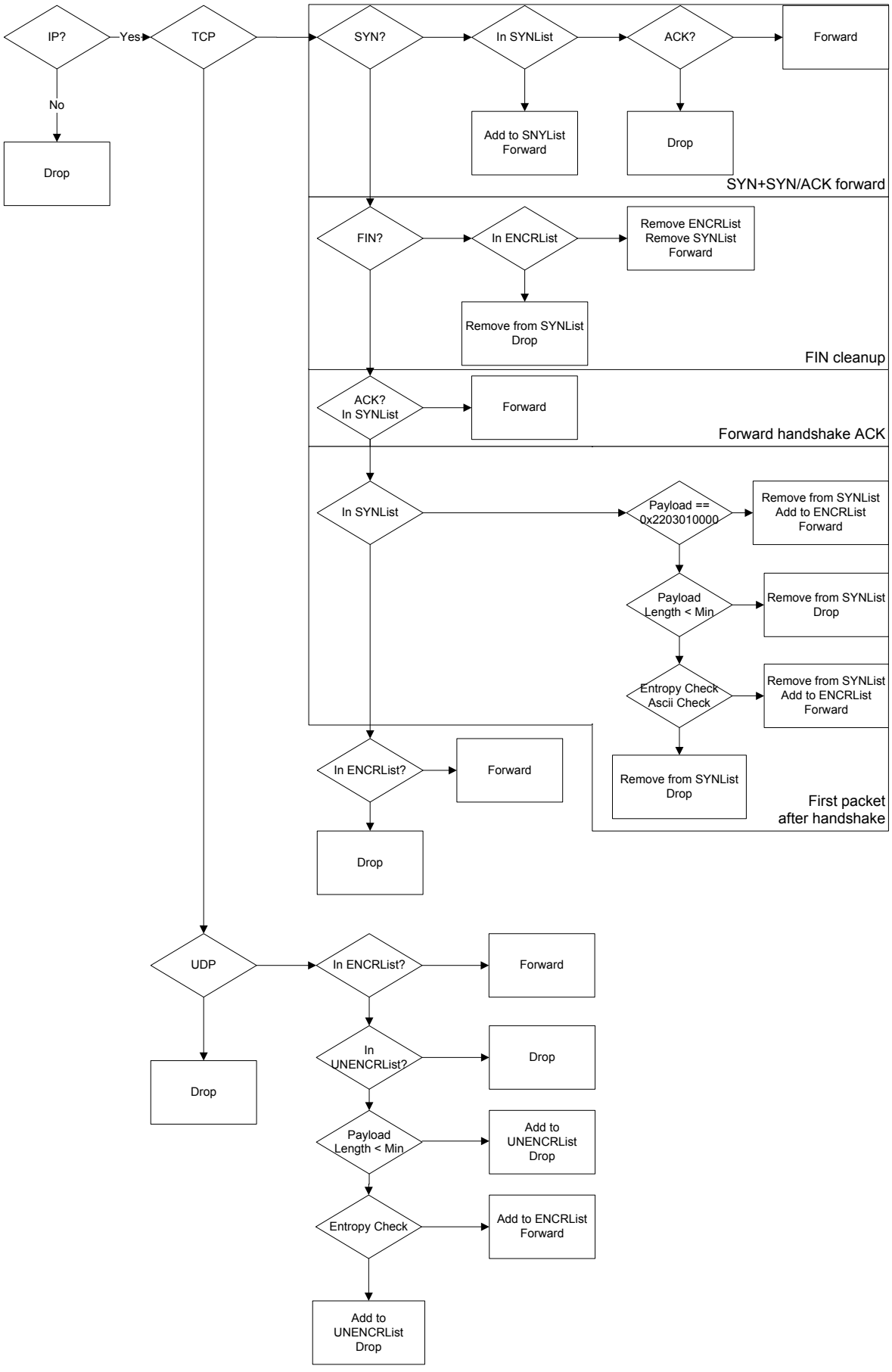


Figure 6.7: SPID pre-filtering flow chart

For this work we will concentrate on Skype, MSE and eDonkey. MSE (Message Stream Encryption) is used by BitTorrent implementations to encrypt the traffic between peers.

It has to be noted, that the definition of Skype flows within SPID is different to the one of the Adami Skype detector. The HTTP Update of Skype is not classified as a Skype flow but as a normal HTTP flow. Likewise HTTPS flows on port 443 are classified as SSL flows. Consequently as pre-filter for SPID to classify Skype, MSE and eDonkey the algorithm will be slightly different. There is no need for the port-based and IP-based classifiers. In Section 6.6.1 we dropped all packets with payload zero, beside the ones of the 3 way handshake, even if the flow was detected as encrypted. For the SPID pre-filter all packets belonging to an encrypted flow are forwarded, independent of the size of the payload.

Figure 6.7 shows the flow chart of the pre-filter for SPID. All non IP traffic is dropped. It is again distinguished between TCP and UDP traffic. The "SYN+SYN/ACK forward" and "FIN cleanup" blocks are the same as for the Skype pre-filter. The "Forward handshake ACK" ensures that the third packet of the 3 way handshake is forwarded. The "First packet after handshake" block, the forthcoming "In ENCRList" check and the corresponding actions are the same as before. Also the UDP filtering is exactly the same as for the Skype pre-filter.

7

Evaluation

This chapter presents evaluation results of the real-time encrypted traffic detector (RT-ETD). The evaluation was performed on five main areas:

1. Influence of the symbol length a,
2. Influence of setting the confidence bound,
3. Coding-based classifier,
4. Pre-filtering of Skype traffic,
5. Pre-filtering for SPID classifier.

For an evaluation of a traffic classifier, traces where the ground truth is known are essential. There are two ways to create such traces:

1. Producing interesting traffic in a controlled environment,
2. Classifying traffic from representative traces.

The former option can be done relatively easy but the quality mainly depends on the composition and preparation of the environment. In most cases the result does not reflect the wide variety of different traffic types and patterns and thus has limited usability. The latter option does not require extensive preparation at the cost of a very complex and time-consuming subsequent analysis. Furthermore, a perfect identification solely based on the traffic itself is very hard or even impossible.

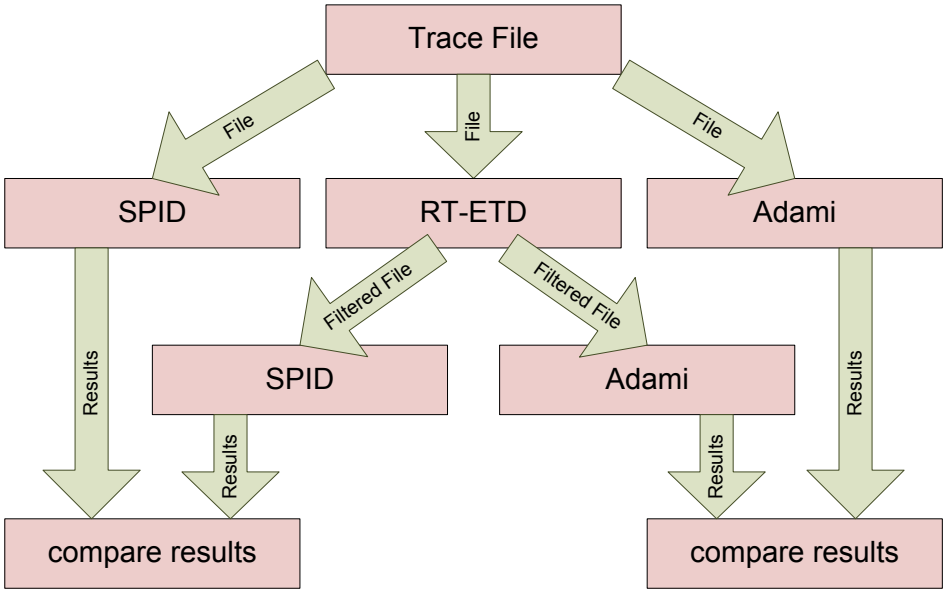


Figure 7.1: Evaluation process

For the evaluation we used real network traces and traces where the ground truth is known. The authors of [26] provided us with some of their fully classified traces of encrypted traffic, which will be referred to as ground truth traces for the rest of the thesis. These traces are used for the evaluation of the parameter settings of the entropy-based classifier. Real network traces have been collected in a network of a small cable network provider in a segment used by about 100 customers, the average traffic volume is about 4Mbit/s. Several traces have been collected in this network for the evaluation we are using three of them. A 1h/2GB trace, 7.5h/13GB trace and a 35h/48GB trace. The traces in this network have been collected in October 2009. Additionally a trace from the network of a wireless provider used by about 1000 customers has been captured in May 2010. The trace covers about 1 hour and has a volume of about 13GB, which is an average rate of about 30Mbit/s. For collecting the traces a mirror port at a fibre switch was used.

Figure 7.1 shows a schematic representation of the evaluation process. In the first step the collected trace files are processed by SPID, the Adami Skype detector and RT-ETD. The results of the Adami Skype detector and SPID are used as 100% within the individual traffic categories. For the filesize the size of the original trace files is used as 100%. The filtered output files from the RT-ETD are then processed by the Adami

as	Skype TCP	Skype UDP	MSE	eDonkey TCP	eDonkey UDP
original	91	1973	649	398	828
1	100%	95.6%	96.6%	96.2%	99%
2	100%	96.7%	98.6%	96.5%	98.4%
3	100%	97.5%	99.4%	97.7%	98.8%
4	100%	97.8%	98.3%	98.2%	99.3%
5	100%	**98.6%**	99.5%	98.2%	99.8%
6	98.9%	**98.6%**	**100%**	**99.0%**	99.4%
7	100%	98.4%	99.5%	**99.0%**	**100%**
8	97.8%	98.0%	99.2%	94.5%	99.6%
9	94.5%	91.0%	99.4%	83.2%	98.7%
10	65.9%	15.3%	99.4%	72.9%	96.4%
11	26.4%	13.3%	99.2%	55.5%	1.4%
12	3.3%	12.0%	60.0%	30.2%	1.2%

Table 7.1: Evaluation of a based on ground truth trace

Skype detector and SPID. The results are then compared to the results based on the original file. The metric we are using here is the fraction of flows in each category that can still be detected in the filtered file and the fraction by which the size of trace file was reduced by the RT-ETD. An optimal result would be if still 100% of the encrypted flows are present in the filtered files, and no unencrypted flows are present.

7.1 Influence of the Symbol Length

As setting the symbol length (a) to an appropriate value is not trivial, an evaluation for a starting from one to twelve is given. To avoid influences by other classifiers, only the entropy-based classifier is used. Flows where the first packet is shorter than $\sqrt{m}$ are dropped. The evaluation is based on the ground truth traces. Table 7.1 shows the number of flows in each category based on the original trace file which is used as 100%. For each a the fraction of detected flows within the filtered file in each category is given. For example 98.9% for $a = 6$ in the Skype TCP category represents that 90 flows out of 91 Skype TCP flows in the trace file are still present in the filtered file. For $a > 8$ the detection ratio decreases as the fraction of packets shorter than $\sqrt{m}$ increase. Especially for encrypted eDonkey UDP traffic $a > 10$ lead to worse results as more than 98% of the first packets are shorter than 52 bytes, and consequently not evaluated.

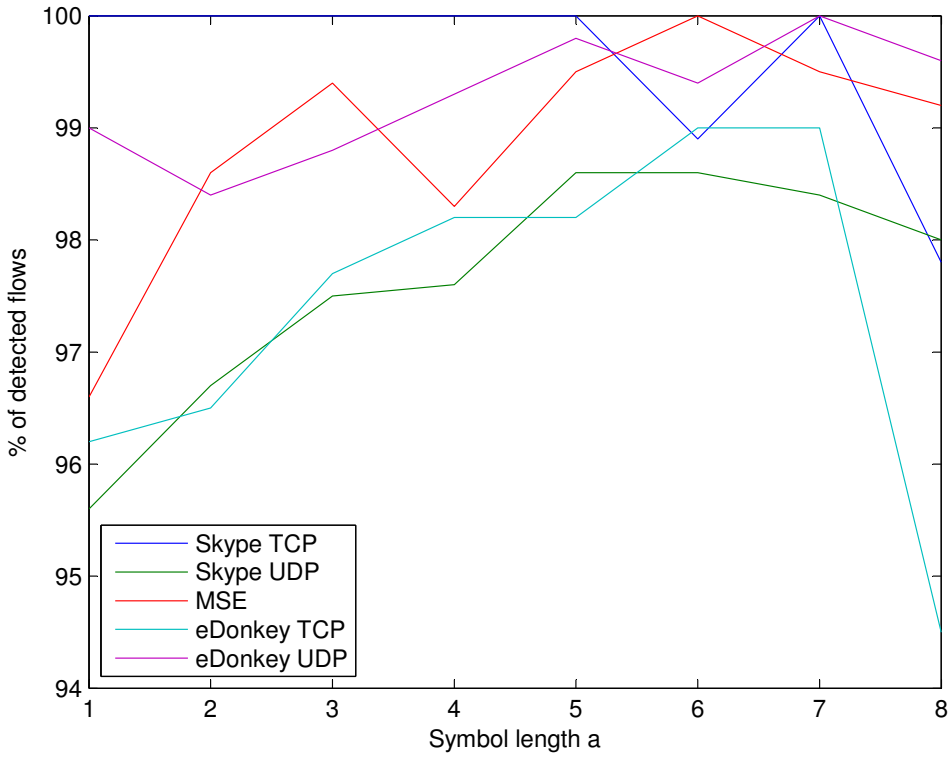


Figure 7.2: Evaluation of a based on ground truth trace

Figure 7.2 plots the fraction of detected flows for a from one to eight. It seems obvious from this figure, that an a of six would be the optimal choice for the detection of this five traffic categories. If our algorithm should be targeted to a specific type of traffic a different a would lead to better results, e.g. for eDonkey UDP an a of seven lead to the best results. As we are only evaluating on encrypted ground truth traces the best results would be performed by an algorithm that simply forwards all packets, what has been neglected in this evaluation is the amount by which the approach removes unencrypted traffic from a trace. We performed an evaluation on a real network trace to have results for the amount of data reduction.

Table 7.2 shows the results for a one hour / 2GB real network trace collected in the cable network. For the detection of Skype flows the Adami Skype detector [1] was used. For MSE and eDonkey traffic SPID [26] was used. SPID does not detect any eDonkey TCP flows, so this column is omitted. For an a of eight the size of the file is reduced by a factor of about fifty. Whereas for an a of six the factor of data

		Skype					eDonkey UDP
a	size	Ping	UDP probe	UDP call	TCP	MSE	encrypted
original	2GB	2840	1445	1	12	4	5
1	7.85%	96.4%	97.2%	100%	91.7%	50%	100%
2	4.11%	97.7%	97.9%	100%	100%	75%	100%
3	4.98%	98.2%	98.7%	100%	100%	75%	100%
4	2.55%	98.6%	**99.7%**	100%	100%	75%	100%
5	7.64%	98.9%	99.6%	100%	100%	75%	100%
6	4.67%	98.8%	**99.7%**	100%	100%	75%	100%
7	9.50%	**99.1%**	99.4%	100%	100%	75%	100%
8	1.99%	99.0%	99.5%	100%	100%	75%	100%
9	6.63%	97.1%	85.1%	100%	100%	75%	80%
10	2.28%	66.5%	1.52%	100%	91.7%	75%	80%
11	2.12%	60.2%	0.04%	100%	41.7%	75%	40%
12	**1.83%**	55.1%	0.04%	0%	8.33%	75%	0%

Table 7.2: Evaluation of a based on real network trace

reduction is in the range of twenty. Based on the ground truth traces the optimal setting would have been an a of six, whereas from a viewpoint of data reduction six is not optimal. Consequently, the appropriate setting of a depends on whether the factor of data reduction or the fraction of forwarded encrypted flows is more important. For $a > 9$ again a lot of encrypted flows are dropped, even if $a = 12$ is the best symbol length concerning data reduction, there is no practical usage as for example almost all Skype UDP probes are removed.

As the approach is targeted to operate in a privacy preserving environment, the factor of data reduction plays a central role. Thus we decided to use an a of eight for the rest of the thesis. What is obvious from Table 7.2 is the bad detection quality for MSE flows. Based on manual deep packet inspection the missing MSE flow seems to be a false positive of SPID, as the remote IP/port pair is hosting a service for a SPAM filter software. Consequently, based on ground truth and real network traces the fraction of encrypted flows that is still present in the filtered file is greater than 94% for all categories.

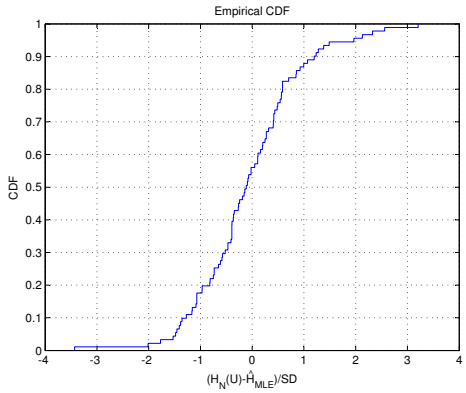
7.2 Appropriate Size of Confidence Bound

Based on a Monte-Carlo method we decided to use a confidence bound of 3 times the standard deviation (see Chapter 6). If $\hat{H}_{\text{MLE}}$ is within $H_N(U) \pm 3 \times SD$ we assume that the flow is encrypted. We will now investigate on this in more detail by evaluating how close flows to this border are. The same traces as in Section 7.1 are used.

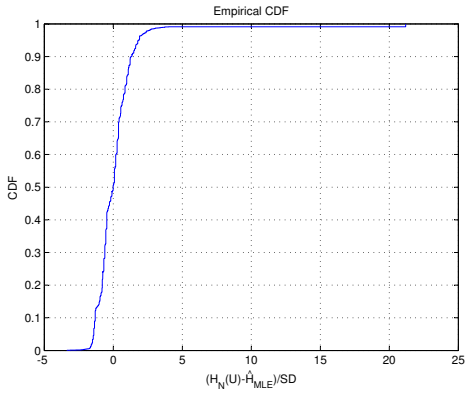
Figure 7.3 shows the CDFs of $\left(H_N(U) - \hat{H}_{\text{MLE}}\right)/SD$ for the 6 different traces. For the encrypted ground truth traces a large fraction of flows is within $3 \times SD$. Based on Figure 7.3 it may be worth having a closer look on the influence of increasing the confidence bound to $4 \times SD$, as a significant fraction of encrypted flows are between $3 \times SD$ and $4 \times SD$. A few flows from encrypted applications are even beyond $18 \times SD$, which cannot be detected by the proposed approach, as increasing the confidence bound this wide, would have the consequence of forwarding about 35% of the flows including a large fraction of unencrypted traffic. As flows where $\hat{H}_{\text{MLE}}$ is greater than $H_N(U) + 3 \times SD$ are very likely to be encrypted the confidence bound could be changed from symmetric to asymmetric. For example to forward all traffic where $\hat{H}_{\text{MLE}}$ is greater than $H_N(U) - 3 \times SD$. For the calculation of the length of the 99% confidence bound we have been using the 0.5% and 99.5% percentile of $H_N(U)$. Again flows where $\hat{H}_{\text{MLE}}$ is greater than the 99.5% percentile are very likely to be encrypted. Consequently also the 0.5% percentile can be used as confidence bound, and we forward all traffic that is beyond this value, this would lead to the aspect that we are 99.5% sure that the encrypted traffic is forwarded.

Table 7.3 shows a comparison between the four different confidence bounds. Concerning the confidence bound of $4 \times SD$, all Skype TCP and eDonkey UDP flows are within $4 \times SD$. For the other three categories the fraction of flows detected as encrypted is slightly increased. Using the other two confidence bounds lead to similar results as $3 \times SD$.

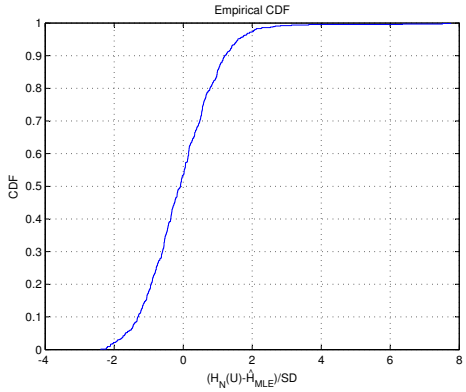
Increasing the confidence bound to $4 \times SD$ will have the consequence to forward more traffic, and consequently not only more encrypted traffic, but also unencrypted traffic will be forwarded. We thus present the same evaluation on our real network trace.



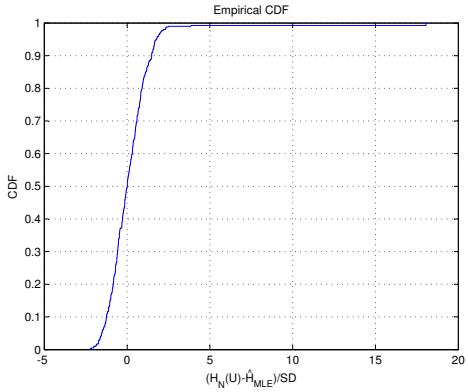
(a) Skype TCP



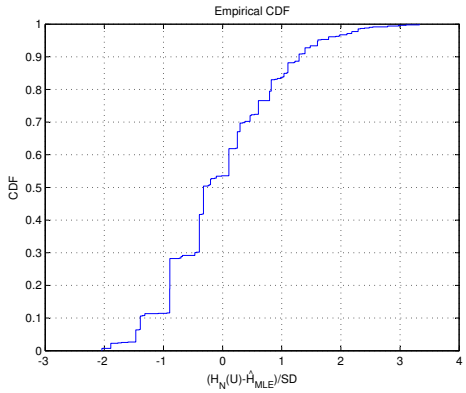
(b) Skype UDP



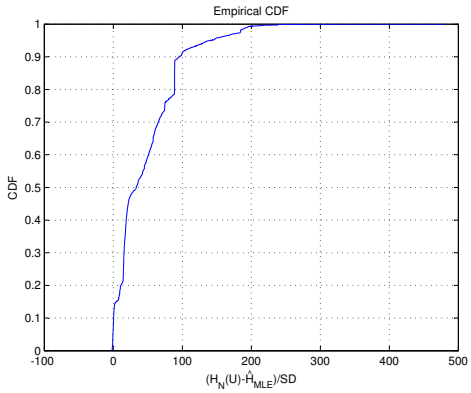
(c) MSE



(d) eDonkey TCP



(e) eDonkey UDP



(f) 1h/2GB trace

Figure 7.3: CDFs of $\left(H_N(U) - \hat{H}_{\mathrm{MLE}}\right)/SD$

conf. bound	Skype TCP	Skype UDP	MSE	eDonkey TCP	eDonkey UDP
original	91	1973	649	398	828
$H_N(U) \pm 3 \times SD$	97.8%	98.1%	99.2%	94.5%	99.6%
$H_N(U) \pm 4 \times SD$	100%	98.7%	99.5%	94.7%	100%
$> H_N(U) - 3SD$	98.9%	98.2%	99.2%	94.5%	99.6%
$> 0.5\%$ percentile	98.9%	98.1%	99.1%	94.7%	99.4%

Table 7.3: Evaluation of confidence bound based on ground truth trace

		Skype					eDonkey
conf. bound	size	Ping	UDP probe	UDP call	TCP	MSE	UDP
original	2GB	2840	1445	1	12	4	5
$H_N(U) \pm 3 \times SD$	1.99%	99.0%	99.5%	100%	100%	75%	100%
$H_N(U) \pm 4 \times SD$	2.08%	99.3%	99.9%	100%	100%	75%	100%
$> H_N(U) - 3SD$	2.02%	99.0%	99.5%	100%	100%	75%	100%
$> 0.5\%$ percentile	2.01%	98.8%	99.7%	100%	100%	75%	100%

Table 7.4: Evaluation of confidence bound based on real network trace

Based on this trace increasing the confidence bound to $4 \times SD$ would result in forwarding 36 additional flows. Based on the Adami Skype detector 16 flows of them are Skype flows. Table 7.4 shows the comparison for the confidence bounds based on the real network trace. The size of the output file increases by only 1.8MByte. For the confidence bounds of $> H_N(U) - 3 \times SD$ and the 0.5% percentile the results are similar to $3 \times SD$. The filesize increases by 600kByte and 400kByte respectively.

The appropriate choice of the confidence bound again depends on what is more important, to forward less unencrypted traffic or to forward a large fraction of encrypted traffic. In case of our real network trace only 36 additional flows are forwarded, when increasing the confidence bound from $3 \times SD$ to $4 \times SD$, where 16 are detected as Skype flows, so the fraction of forwarded unencrypted flows is not significantly increased by increasing the confidence bound to $4 \times SD$. For the ongoing work we decided to stay at a confidence bound of $3 \times SD$, as there are almost no practical implications when using $> H_N(U) - 3 \times SD$ and the 0.5% percentile instead. For the confidence bound of $4 \times SD$ a more detailed analysis based on our pre-filter for SPID is given in Section 7.5.

7.3 Evaluation of Coding-based Classifier

Based on the real network trace we evaluated the usage for the coding-based classifier for TCP and UDP traffic. Using the coding-based classifier for UDP traffic does not influence the classification at all. Consequently the coding-based classifier is not used for UDP traffic.

For TCP traffic the coding-based classifier removes about 900kByte from the output file without changing the classification results of the Adami classifier and SPID. 900kByte seems to be a negligible amount of traffic but deep packet inspection revealed that these are POP3 flows where the username and password are transported in plain text, thus the content-based classifier is of great value in this case, as privacy sensitive information is removed.

Until now only the entropy-based and the coding-based classifier have been evaluated, we now continue to evaluate the code proposed in Subsection 6.6.1 to have a Skype pre-filter based on information gathered from the first packet of the flow.

7.4 Skype Traffic Pre-Filter

In this section the algorithm proposed in Section 6.6.1 is evaluated. The evaluation is based on three different real network traces. A 7.5 hours and a 35 hours trace file collected in the cable network and a one hour trace file collected in the wireless network. Table 7.5 shows the results of the evaluation. For the three traces the filesize is reduced by a factor between seven and twenty-five, where the factor for the wireless trace is the smallest. The number of TCP flows is reduced to about 10%. The percentage of UDP flows that is dropped is smallest for the 7.5 hour trace. It has to be noticed that only 20% of the flows in the original file are detected to be non Skype flows, consequently the reduction of about 14% is close to optimal.

For all trace files the fraction of Skype flows that is still present is at least 97.6%. Consequently, we are able to drop a large fraction of non Skype flows, and ensure that at least 97.6% Skype flows are still present. It has to be noticed that using the Adami

	7.5h/13GB		35h/48GB		1h/13GB	
Type	original	filtered	original	filtered	original	filtered
Filesize [MB]	13531	5.42%	48527	3.98%	13157	13.8%
TCP flows	143867	11.3%	298856	9.99%	76199	19.0%
UDP flows	35076	85.9%	494489	8.85%	80568	49.9%
Skype TCP	249	98.4%	243	98.8%	255	98.4%
Sykpe UDP	28204	99.0%	39394	98.7%	9512	97.6%

Table 7.5: Results for Skype pre-filter

Skype detector to process the original file takes significantly longer than pre-filtering the original file and then processing the filtered result.

7.5 SPID Traffic Pre-Filter

The evaluation of the algorithm proposed in Section 6.6.2 is based on the same trace files as the evaluation of the Skype traffic pre-filter. Table 7.6 shows the results of the evaluation. A zero indicates a 0 count, this category is completely removed, whereas 0% indicates that the fraction is below 0.01%.

The first major difference is that the filtered files are about 25% smaller than the pre-filtered files for Skype detection. The major part of the difference is HTTPS traffic on port 443, as for Skype pre-filtering we forward all TCP traffic on port 443.

First we concentrate on filtering with the confidence bound of $3 \times SD$. Between 73% and 96% of the flows are dropped by our pre-filter. Unencrypted flows such as FTP, HTTP, IMAP or SMTP are almost completely removed, which is a strong indication that only a small fraction of unencrypted traffic is forwarded. For encrypted protocols that we take into account, eDonkey TCP/UDP encrypted, MSE, Skype TCP/UDP at least 76.7% (MSE) of the flows are still present in the filtered file. For eDonkey and Skype the fraction is above 93%. What is obvious is the difference between the number of Skype flows detected by SPID and the Adami Skype detector. For UDP the major reason is the aspect that SPID does not use a timeout while Adami does.

For TCP it highlights the aspect that Skype detection is not trivial, and consequently two different algorithms lead to different results when executed on the same file. For example the signature of Adami's Skype Login message is HEX "2203010000", in the

Type	7.5h/13GB			35h/48GB			1h/13GB		
	original	filtered $3 \times SD$	filtered $4 \times SD$	original	filtered $3 \times SD$	filtered $4 \times SD$	original	filtered $3 \times SD$	filtered $4 \times SD$
Filesize [MB]	13531	3.36%	3.37%	48527	2.94%	3.15%	13157	12.5%	12.8%
Sessions	242309	13.0%	13.1%	1050206	4.37%	4.42%	195243	27.1%	27.5%
BitTorrent	64	0	0	8067	0	0	5061	0	0
DNS	30946	0	0	91015	0	0	22166	0	0
eDonkey	0	0	0	0	0	0	7169	0	0
eDonkeyTCP encr.	9	100%	100%	36	100%	100%	9653	96.2%	96.6%
eDonkeyUDP encr.	44	93.2%	95.5%	95	100%	100%	21808	96.0%	96.3%
FTP	443	0	0	31	0	0	24	0	0
HTTP	118226	0	0	210320	0%	0%	46643	0	0%
IMAP	248	0	0	1861	0	0	26	0	0
IRC	0	0	0	68	0	0	66	0	0
ISAKMP	4	0	0	3	33%	33%	227	0	0
MSE	30	76.7%	76.7%	280	99.3%	100%	1323	81.7%	81.9%
MSN	152	0	0	19	0	0	23	0	0
POP	9770	0	0	25528	0	0	632	0	0
SkypeTCP	773	99.5%	100%	1167	99.4%	99.8%	456	94.1%	95.0%
SkypeUDP	18945	99.0%	99.5%	27675	99.2%	99.7%	6079	98.7%	100%
SMTP	832	0	0	1075	0	0	53	0	0
SpotifyServer	55	74.5%	74.5%	90	94.4%	94.4%	306	95.4%	96.1%
SSH	946	0	0	60529	0	0	26	0	0
SSL	10044	0	0	19203	0%	0.18%	3210	0	0%
UNKNOWN	50778	23.3%	23.6%	603144	2.8%	2.8%	70292	21.2%	22.0%

Table 7.6: Results for SPID pre-filter

ground truth files from the authors of [26] the Skype Login messages have a payload of "1603010000". Consequently, the Adami Skype detector fails to detect these Skype flows.

Increasing the confidence bound to $4 \times SD$ slightly increase filesize as well as detection quality. As stated above there is a trade-off between detection performance and privacy preservation.

8

Conclusion

In this thesis a real-time encrypted traffic detector (RT-ETD) is proposed. The RT-ETD gathers information from the first packet of a flow, and decides whether the flow is encrypted or not.

The work is motivated by the evolvement of privacy legislation into the digital life, and the need for appropriate traffic classification to perform QoS. The RT-ETD can be used as a privacy preserving pre-filter of network traffic. The pre-filtered traffic is then used in more detailed classification steps.

The task of the RT-ETD is to gather information from the first packet of a flow and decide whether the flow is encrypted or not. The core concept of the approach is based on estimating the entropy of the payload. This estimated value is then compared to the estimated entropy of a uniformly distributed payload of the same length. Based on the 99% confidence interval it is then decided whether the flow is encrypted or not.

The algorithm of the RT-ETD is enhanced by the usage of further classification modules. Within these modules the decision is based on port numbers, IP addresses, statistical information about the payload or the content of the payload.

The RT-ETD advances current state of the art in three main areas:

1. Improved user privacy, due to the usage of the RT-ETD as pre-filter in privacy preserving network monitoring,

2. Proposal of a new metric in traffic classification based on entropy estimation of the payload of the first packet of a flow,

3. Real-Time traffic pre-filtering as only processing information from the first packet of a flow.

For protecting user privacy the RT-ETD is best used in a privacy preserving network monitoring architecture (e.g. PRISM [75]), which provides further modules for privacy protection such as anonymization or authorization.

The RT-ETD is evaluated by the usage of ground truth traces and real network traces. Based on the ground truth traces it is shown that more than 94% of the encrypted flows are detected as encrypted. Based on real network traces it is shown that more than 90% of the traffic is dropped. These two values are mainly influenced by setting two parameters of the RT-ETD, and it can thus be decided whether data reduction or recall is more important.

Finally it seems promising to use entropy of the payload of the first packet as a metric for the likeliness of encryption of a flow. This metric is able to enhance the quality of state of the art traffic classifiers like SPID [26].

8.1 Future Work

In the future the work will be enhanced by a theoretical analysis to presume the amount of unencrypted traffic that falls within our given confidence interval. For the practical evaluation of this a ground truth trace file with unencrypted traffic will be collected.

Further the presented work will be enhanced such that the RT-EDT is able to operate in a high-speed multi-layer traffic classification algorithm, to be implemented within future firewall systems. There the concepts of the RD-EDT will be used to provide only traffic, that need layer seven approaches for classification, to the layer seven classifier module.

Bibliography

[1] Adami, D., Callegari, C., Giordano, S., Pagano, M., and Pepe, T.: *A Real-Time Algorithm for Skype Traffic Detection and Classification.* In Balandin, S., Moltchanov, D., and Koucheryavy, Y. (editors): *Smart Spaces and Next Generation Wired/Wireless Networking*, pages 168–179. Springer-Verlag, Berlin, Heidelberg, 2009.

[2] Antos, A. and Kontoyiannis, I.: *Convergence properties of functional estimates for discrete distributions.* Random Struct. Algorithms, 19(3-4):163–193, 2001.

[3] Bennett, J. and Zhang, H.: *WF2Q: Worst-case Fair Weighted Fair Queueing.* In *Procedings of the IEEE INFOCOM'96*, page pp.120, San Francisco, CA, 1996.

[4] Berger, A., Della Pietra, V., and Della Pietra, S.: *A maximum entropy approach to natural language processing.* Comput. Linguist., 22(1):39–71, 1996.

[5] Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., and Tofanelli, P.: *Revealing Skype traffic: when randomness plays with you.* SIGCOMM Comput. Commun. Rev., 37(4):37–48, 2007.

[6] Callado, A., Szabó, C., Gero, B., Kelner, J., Fernandes, S., and Sadok, D.: *A survey on internet traffic identification.* IEEE Communications Surveys & Tutorials, 11(3):37–52, 2009.

[7] Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, C.: *RFC 1157: Simple Network Management Protocol (SNMP).* Internet Engineering Task Force, 1990.

[8] Cho, K., Fukuda, K., Esaki, H., and Kato, A.: *Observing slow crustal movement in residential user traffic.* In Azcorra, Arturo, Veciana, Gustavo de, Ross, Keith W., and Tassiulas, Leandros (editors): *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12. ACM, New York, NY, 2008.

[9] Choi, T., Kim, C., Yoon, J., Park, J., Lee, B., Kim, H., Chung, H., and Jeong, T.: *Content-aware internet application traffic measurement and analysis.* In *Proceedings of Network Operations and Management Symposium*, pages 511–524. IEEE Computer Society, Washington, DC, 2004.

[10] Cisco Systems: *Netflow - cisco systems.* `http://www.cisco.com/en/US/tech/tk812/tsd_technology_support_protocol_home.html` (20.08.2010).

[11] Cover, T. M. and Thomas, J. A.: *Elements of information theory.* Wiley-Interscience, New York, NY, 1991.

[12] Crispin, M.: *RFC 3501: INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.* Internet Engineering Task Force, 2003.

[13] Crotti, M., Dusi, M., Gringoli, F., and Salgarelli, L.: *Traffic classification through simple statistical fingerprinting.* SIGCOMM Comput. Commun. Rev., 37(1):5–16, 2007.

[14] Dainotti, A., Donato, W. de, Pescapè, A., and Rossi, P. S.: *Classification of network traffic via packet-level hidden markov models.* In *Proceedings of the Global Communications Conference GLOBECOM*, pages 2138–2142. IEEE Computer Society, Washington, DC, 2008.

[15] DasGupta, A.: *The matching, birthday and the strong birthday problem: a contemporary review.* Journal of Statistical Planning and Inference, 130(1-2):377–389, 2005.

[16] Dorfinger, P., Brandauer, C., and Hofmann, U.: *A rate controller for long-lived tcp flows.* In Boavida, Fernando, Monteiro, Edmundo, and Orvalho, João (editors): *IDMS/PROMS*, volume 2515 of *Lecture Notes in Computer Science*, pages 154–165. Springer, Berlin, Heidelberg, 2002.

[17] Dougherty, E. R. and Laplante, P. A.: *Introduction to Real-Time Imaging.* Wiley-IEEE Press, Piscataway, NJ, 1995.

[18] Duda, R. O., Hart, P. E., and Stork, D. G.: *Pattern Classification (2nd Edition).* Wiley-Interscience, New York, NY, 2001.

[19] Efron, B. and Stein, C.: *The jackknife estimate of variance.* Annals of Statistics, 9(3):586–596, 1981.

[20] Erman, J., Arlitt, M., and Mahanti, A.: *Traffic classification using clustering algorithms.* In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM, New York, NY, 2006.

[21] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T.: *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1.* Internet Engineering Task Force, 1999.

[22] Friedman, W. F.: *The index of coincidence and its applications in cryptology.* Riverbank Laboratories, Geneva, IL, 1920.

[23] Gaudino, F., Schmidl, M., Rittweger, C., Lebeau-Marianna, D., Quoy, N., Charlot, D., Story, C., Saltzman, I., Walden, I., Passadelis, N., Spring, P., Boschi, E., Rao, S., Lioudakis, G., Kaklamani, D., and Venieris, I.: *D2.1.1: Assessment of the legal and regulatory framework.* `http://fp7-prism.eu/images/upload/Deliverables/fp7-prism-wp2.1-d2.1.1-v1.pdf` (20.08.2010), 2008.

[24] Gray, R. M.: *Entropy and Information Theory.* Springer-Verlag, New York, NY, 2009. `http://ee.stanford.edu/~gray/it.html(20.08.2010)`.

[25] Haffner, P., Sen, S., Spatscheck, O., and Wang, D.: *Acas: automated construction of application signatures*. In Sen, Subhabrata, Ji, Chuanyi, Saha, Debanjan, and McCloskey, Joe (editors): *MineNet*, pages 197–202. ACM, New York, NY, 2005.

[26] Hjelmvik, E. and John, W.: *Statistical protocol identification with spid: Preliminary results*. In *SNCNW'09: Swedish National Computer Networking Workshop*, Uppsala, 2009.

[27] IANA: *Port numbers*. `http://www.iana.org/assignments/port-numbers` (20.08.2010).

[28] Iliofotou, M., Kim, H., Faloutsos, M., Mitzenmacher, M., Pappu, P., and Varghese, G.: *Graph-based p2p traffic classification at the internet backbone*. In *INFOCOM'09: Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, pages 37–42. IEEE Computer Society Press, Piscataway, NJ, 2009.

[29] IPOQUE: *Internet study 2007*. `http://www.ipoque.com/userfiles/file/ipoque-internet-study-2007-abstract-en.pdf` (20.08.2010), 2007.

[30] John, W.: *Characterization and Classification of Internet Backbone Traffic*. Chalmers Reproservice, Goteburg, 2010.

[31] John, W. and Tafvelin, S.: *Heuristics to classify internet backbone traffic based on connection patterns*. In *ICOIN '08: Proceedings of the 22nd International Conference on Information Networking*. IEEE Computer Society Press, Piscataway, NJ, 2008.

[32] Jordan, S.: *Four questions that determine whether traffic management is reasonable*. In *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, pages 137–140. IEEE Computer Society Press, Piscataway, NJ, 2009.

[33] Jordan, S.: *Implications of internet architecture on net neutrality*. ACM Trans. Internet Technol., 9(2):1–28, 2009.

[34] Karagiannis, T., Broido, A., Faloutsos, M., and claffy, kc: *Transport layer identification of p2p traffic*. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134. ACM, New York, NY, 2004.

[35] Karagiannis, T., Papagiannaki, K., and Faloutsos, M.: *Blinc: multilevel traffic classification in the dark*. In Guérin, R., Govindan, R., and Minshall, G. (editors): *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 229–240. ACM, New York, NY, 2005.

[36] Kim, H., Claffy kc, Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K.: *Internet traffic classification demystified: myths, caveats, and the best practices*. In Azcorra, Arturo, Veciana, Gustavo de, Ross, Keith W., and Tassiulas, Leandros (editors): *Proceedings of the 2008 ACM Conference on Emerging Network Experiment and Technology, CoNEXT*, pages 1–12. ACM, New York, NY, 2008.

[37] Klensin, J.: *RFC 5321: Simple Mail Transfer Protocol*. Internet Engineering Task Force, 2008.

[38] Kullback, S. and Leibler, R. A.: *On information and sufficiency*. Annals of Mathematical Statistics, 22(1):79–86, 1951.

[39] Leibowitz, N., Ripeanu, M., and Wierzbicki, A.: *Deconstructing the Kazaa network*. In *WIAPP '03: Proceedings of the Third IEEE Workshop on Internet Applications*, pages 112–120. IEEE Computer Society, Washington, DC, 2003.

[40] Lewen, U. M.: *Internet File-Sharing: Swedish Pirates Challenge the U.S.* Cardozo J. Int'l & Comp. L., 16(1):173–205, 2008.

[41] Lu, L., Horton, J., Safavi-Naini, R., and Susilo, W.: *Transport layer identification of Skype traffic*. In *ICOIN'07*, volume 5200 of *Lecture Notes in Computer Science*, pages 465–481. Springer, Berlin, Heidelberg, 2008.

[42] Lyda, R. and Hamrock, J.: *Using entropy analysis to find encrypted and packed malware*. IEEE Security & Privacy, 5(2):40–45, 2007.

[43] Ma, J., Levchenko, K., Kreibich, C., Savage, S., and Voelker, G. M.: *Unexpected means of protocol inference*. In Almeida, J. M., Almeida, V. A. F., and Barford, P. (editors): *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement 2006*, pages 313–326. ACM, New York, NY, 2006.

[44] Mackay, D. J. C.: *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, Cambridge, 1st edition, 2002.

[45] Maier, G., Feldmann, A., Paxson, V., and Allman, M.: *On dominant characteristics of residential broadband internet traffic*. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 90–102. ACM, New York, NY, 2009.

[46] Maiolini, G., Baiocchi, A., Rizzi, A., and Di Iollo, C.: *Statistical classification of services tunneled into ssh connections by a k-means based learning algorithm*. In *IWCMC '10: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 742–746. ACM, New York, NY, 2010.

[47] Menezes, A., Vanstone, S., and Oorschot, P.: *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, 1996.

[48] Miller, G.: *Note on the bias of information estimates*. Information theory in psychology, II-B:95–100, 1955.

[49] Mills, D., Martin, J., Burbank, J., and Kasch, W.: *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification*. Internet Engineering Task Force, 2010.

[50] Mochalski, K. and Schulze, H.: *Deep packet inspection: Technology, applications & net neutrality*. Technical report, IPOQUE, 2009. `http://www.csn-germany.de/download/Whitepaper_DPI.pdf` (20.08.2010).

[51] Mockapetris, P.V.: *RFC 1034: Domain names - concepts and facilities*. Internet Engineering Task Force, 1987.

[52] Moore, A. W. and Papagiannaki, K.: *Toward the accurate identification of network applications*. In Dovrolis, C. (editor): *PAM: Passive and Active Network Measurement Workshop*, volume 2515 of *Lecture Notes in Computer Science*, pages 41–54. Springer, Berlin, Heidelberg, 2005.

[53] Myers, J. and Rose, M.: *RFC 1939: Post Office Protocol - Version 3*. Internet Engineering Task Force, 1996.

[54] Nguyen, T. T. T. and Armitage, G. J.: *A survey of techniques for internet traffic classification using machine learning*. IEEE Communications Surveys and Tutorials, 10(1-4):56–76, 2008.

[55] Olivain, J. and Goubault-Larrecq, J.: *Detecting subverted cryptographic protocols by entropy checking*. Research report LSV-06-13, Laboratoire Spécification et Vérification, ENS Cachan, 2006. `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2006-13.pdf(20.08.2010)`.

[56] Paninski, L.: *Estimation of information-theoretic quantities*. `http://www.stat.columbia.edu/~liam/research/info_est.html` (20.08.2010).

[57] Paninski, L.: *Estimation of entropy and mutual information*. Neural Computation, 15(6):1191–1253, 2003.

[58] Paninski, L.: *Estimating entropy on m bins given fewer than m samples*. IEEE Transactions on Information Theory, 50(9):2200–2203, 2004.

[59] Paninski, L.: *A coincidence-based test for uniformity given very sparsely sampled discrete data*. IEEE Transactions on Information Theory, 54(10):4750–4755, 2008.

[60] Pescape, A.: *Entropy-based reduction of traffic data*. IEEE Communications Letters, 11(2):191–193, 2007.

[61] Postel, J.: *RFC 768: User Datagram Protocol*. Internet Engineering Task Force, 1980.

[62] Postel, J.: *RFC 793: Transmission Control Protocol*. Internet Engineering Task Force, 1981.

[63] Postel, J. and Reynolds, J. K.: *RFC 959: File Transfer Protocol*. Internet Engineering Task Force, 1985.

[64] Rish, I.: *An empirical study of the naive bayes classifier*. In *IJCAI-01 workshop on "Empirical Methods in AI"*, pages 41–46, Seattle, WA, 2001.

[65] Sahu, S., Nain, P., Towsley, D., Diot, C., and Firoiu, V.: *On Achievable Service Differentiation with Tocken Bucket Marking for TCP*. In *Proc. of the ACM SIGMETRICS'2000 Int. Conf. on Measurement and Modeling of Computer Systems*, pages 23–33, Santa Clara, CA, 2000.

[66] Schapire, R. E.: *The boosting approach to machine learning: An overview.* In D. Denison, M. Hansen, C. Holmes B. Mallick B. Yu (editor): *MSRI Workshop on Nonlinear Estimation and Classification*. Springer, New York, NY, 2002.

[67] Schneider, P.: *TCP/IP Traffic Classification Based on Port Numbers.* `http://www.schneider-grin.ch/media/pdf/diploma_thesis.pdf` (20.08.2010), 1996.

[68] Schürmann, T.: *Bias analysis in entropy estimation.* Journal of Physics A: Mathematical and General, 37(27):L295–L301, 2004.

[69] Sen, S., Spatscheck, O., and Wang, D.: *Accurate, scalable in-network identification of p2p traffic using application signatures.* In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 512–521. ACM, New York, NY, 2004.

[70] Shannon, C. E.: *A mathematical theory of communication.* Bell System Technical Journal, 27:379–423, 625–56, 1948.

[71] Skype Limited: *Skype - make free calls and great value calls on the internet.* `http://www.skype.com` (20.08.2010).

[72] Strong, S. P., Koberle, R., Steveninck, R. R. de Ruyter van, and Bialek, W.: *Entropy and information in neural spike trains.* Phys. Rev. Lett., 80(1):197–200, 1998.

[73] Svoboda, P., Hyytiä, E., Ricciato, F., Rupp, M., and Karner, M.: *Detection and tracking of Skype by exploiting cross layer information in a live 3G network.* In Papadopouli, M., Owezarski, P., and Pras, A. (editors): *TMA*, volume 5537 of *Lecture Notes in Computer Science*, pages 93–100. Springer, Heidelberg, Berlin, 2009.

[74] Szabó, G., Orincsay, D., Malomsoky, S., and Szabó, I.: *On the validation of traffic classification algorithms.* In Claypool, M. and Uhlig, S. (editors): *PAM*, volume 4979 of *Lecture Notes in Computer Science*, pages 72–81. Springer, Heidelberg, Berlin, 2008.

[75] Telscom AG: *Privacy-aware secure monitoring.* `http://fp7-prism.eu` (20.08.2010).

[76] The MathWorks: *Matlab documentation.* `http://www.mathworks.com/access/helpdesk/help/techdoc/` (20.08.2010).

[77] Trammel, B., Boschi, E., Procissi, G., Callegari, C., Dorfinger, P., and Schatzmann, D.: *A long history of skype: identifying and exploring skype traffic in a large-scale, long-term flow data repository*, 2010. Submitted for Publication.

[78] Wagner, A. and Plattner, B.: *Entropy based worm and anomaly detection in fast IP networks.* In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 172–177. IEEE Computer Society, Washington, DC, 2005.

[79] Wondracek, G., Holz, T., Kirda, E., and Kruegel, C.: *A practical attack to de-anonymize social network users*. In *IEEE Symposium on Security and Privacy*, pages 223–238. IEEE Computer Society, Washington, DC, 2010.

[80] Yaghmour, K., Masters, J., Ben-Yossef, G., and Gerum, P.: *Building Embedded Linux Systems*. O'Reilly, Sebastopol, CA, 2008.

[81] Ylonen, T. and Lonvick, C.: *RFC 4251: The Secure Shell (SSH) Protocol Architecture*. Internet Engineering Task Force, 2006.

[82] Zhang, M., John, W., claffy, kc, and Brownlee, N.: *State of the art in traffic classification: A research review*, 2009. PAM'09: 10th International Conference on Passive and Active Measurement, Student Workshop, Seoul.

List of Abbreviations

ACK Acknowledgement

ANSI American National Standards Institute

ASCII American Standard Code for Information Interchange

CDF Cumulative Distribution Function

CORBA Common Object Request Broker Architecture

DDM Distributed Data Management

DNS Domain Name Service

FIN Final

FTP File Transfer Protocol

FTPS File Transfer Protocol Secure

HEX Hexadecimal

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDSs Intrusion Detection Systems

IEEE Institute of Electrical and Electronics Engineers

IIOP Internet Inter Object Request Protocol

IMAP Internet Message Access Protocol

IMAPS Internet Message Access Protocol Secure

IP Internet Protocol

IPFIX Internet Protocol Flow Information Export

IRCS Internet Relay Chat Secure

ISPs Internet Service Providers

LDAPS Lightweight Directory Access Protocol Secure

ML Maximum Likelihood

MLE Maximum Likelihood Estimator

MMS Media Management System

MSE Message Stream Encryption

NNTPS Network News Transfer Protocol Secure

NSIIOPS Name Service Internet Inter Object Request Broker Protocol Secure

NTP Network Time Protocol

P2P Peer to Peer

POP Post Office Protocol

POP3 Post Office Protocol Version 3

POP3S Post Office Protocol Version 3 Secure

PRISM Privacy Preserving Secure Network Monitoring

PSH Push

QoS Quality of Service

RT-ETD Real-Time Encrypted Traffic Detector

SD Standard Deviation

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SPID Statistical Protocol Identification

SSH Secure Shell

SSL Secure Socket Layer

SYN Synchronisation

TCP Transmission Control Protocol

TELNETS Telecommunication Network Secure

TSTAT TCP Statistic and Analysis Tool

TLS Transport Layer Security

UDP User Datagram Protocol

VoIP Voice over Interent Protocol

WWW World Wide Web