POLITECNICO DI TORINO

III Facoltà di Ingegneria dell'informazione

Corso di Laurea in Ingegneria telematica

Tesi di Laurea Magistrale

Statistical classification of TCP traffic with Support Vector Machine

Relatore:

prof. Marco Mellia

Correlatore:

prof. Dario Rossi

Candidato:

Gianluca La Mantia

Anno accademico 2007-2008

Alla mia famiglia, per avermi sempre sostenuto e dato coraggio nei momenti piú difficili.

A quella bimba che, con la sua rara dolcezza, ha dato colore e forza a questi cinque combattuti anni.

Queste fatiche sono a voi dedicate.

Summary

Over the last few years traffic on the Internet has increased dramatically, both in terms of amount of traffic and on the variety of different applications. The introduction of voice, video and other real-time applications has changed the way the Internet is used and this has triggered the need for a change in traffic handling and in particular an increasing demand for service differentiation. In fact accurate differentiation and categorization of network traffic according to application type is an important element related to many network management tasks such as flow prioritization, traffic shaping/policing, and diagnostic monitoring. Besides these QoS issues, other activities which would greatly benefit from a traffic characterisation knowledge are anomaly/intrusion detection and high performance networks design.

For these various purposes a classification of services is demanded, but with the ever increasing number of protocols and services running on non-standard transport-layer ports (e.g. tunnelling operations), every method based on the 4^{th} layer header analysis is rapidly becoming ineffective. On the other hand, identification methods consisting of a full payload analysis cannot be largely exploited since they are very computationally expensive and they require pattern matching capabilities, needing to scan all data carried by packets. An example of these old classification systems is *packet inspectors*, analysis automaton based on finite state machines in order to describe protocol header data and which require to hold every protocol dynamics knowledge which is usually described in specific service documentation. Besides the disadvantage of performing pattern matching techniques that need to scan all the data carried by packets, this approach cannot clearly apply to encrypted traffic and not even to proprietary protocols since full documentation is not available and therefore finite state machines cannot be instructed.

For these reasons new approaches have been developed, based on the statistical analysis of different traffic properties, communication patterns and hosts behaviours:

a set of protocol dependent features can be extracted from the results of the packet-level traffic characterization to build a blind classifier that does not require resource intensive processing for packet inspection; it does not need to track protocol variants and evolutions; it is applicable to encrypted traffic and finally it could identify whether a protocol is used natively or as a transport layer for other application protocols.

The work on this thesis follows the previous idea and it focuses on a particular statistical classification model based on the χ^2 goodness of fit test as a classification operator for TCP traffic. In fact it has been shown that if computed over groups of bits the χ^2 test renders a variability degree which can be used to determine if those are used as counters, constant or random values. In this way it is possible to determine the application protocol format and use this feature to perform traffic classification without having to care about specific values of the header fields, proving that just a measure of the bit randomness is enough to permit an exact identification. In particular, in this work, groups of bits are extracted from the TCP payload and, after computations, a vector of χ^2 values is obtained, one for each considered group of bits (they are here called *chunks*). In order to put this concept into practice a specific module for Tstat software has been implemented after modifying an already existing version designed for UDP traffic. This new module, starting from an input trace, is able to extract chunks of bits from the first bytes of TCP packet payloads and compute over them vectors of statistics with a fixed and a beforehand established periodicity. Signatures are produced only after a certain number of packets have been received, as opposed to packet inspectors which can analyze and identify single packets, this means classification cannot be performed immediately for any packets. In fact the basic idea is to store chunks extracted from packets belonging to the same flow, that is to say having the same source and destination hosts and to compute over them statistics after having received a sufficient amount of samples/packets. But since the object of this study is TCP traffic and the classification of its upper layer applicative protocol, it is clear that the kind of flows analyzed are not so voluminous and the periodicity threshold represents a huge obstacle to produce signatures. For this kind of traffic the only solution is to compute statistics after having aggregated packets by endpoints (source or destination), while flow aggregation is not possibile since it is very difficult here to reach a sufficient number of packets for any flow. After this kind of aggregation any signature achieves the meaning of a packet format statistical description produced after having analyzed an amount of packets directed to (or outgoing from) a given endpoint, and this can be also rendered as a transmission study related to a single direction where a given endpoint is involved.

Information regression is committed to a Support Vector Machine classifier which maps the χ^2 vectors as points into a higher dimensional space. In this way the classification problem is easier to solve since samples belonging to different protocols can be well separated in the cartesian space by an hyper-plane with an high discriminating power. From a geometrical point of view the problem results in the identification, after having performed a transform from the original sample space to a new high-dimensional features space, of a portion maximizing the distance between different datapoint clouds and then placing in the middle of this margin a separation plane providing samples classification. As some efficient implementations of this algorithm are already available, LIBSVM library has been adopted, since it provides powerful tools for training and prediction actions and optimization methods such as grid operation or scaling methods.

Once selected the χ^2 as the exploited feature to perform protocol identification, the whole classification model has been designed and implemented. After a trace pre-filtering performed in order to instruct the classifier, the protocol traces are processed by the Tstat TCP-chunker and signatures vectors representing the whole dataset are produced. After a careful sampling the dataset is splitted into a training set and testing set: the first one feeds the SVM in its learning phase and allows it to produce the model used to classify testing objects. Finally, the assigned labels veracity is checked and a performances evaluation is rendered in terms of True Positives, False Positives and accuracy rate.

During the testing phase several aspects have been considered in order to elaborate an effective system setup. After having selected the application protocols for identification, a further background class has been taken into account in order to isolate all traffic not belonging to the given services. First of all a good pre-filtering policy has proved to be decisive for obtaining a reliable classification model, so a sequence of filtering operations based on a Deep Packet Inspection mechanism has been performed on the test trace. Then exhaustive tests about training set dimensioning have been executed and the most effective samples distribution has been a balanced one, built up in a progressive way and avoiding to obtain classes without a sufficient amount of training samples or, on the other hand, classes that are so largely trained that they absorb other classes' samples. Other tests have been performed by changing the number of packets per flow to analyze, or by inserting in any signature 5 more chunks containing all TCP flags and advanced options contained in TCP header. In fact the idea was that by also considering these bits the classification could gain in effectiveness, but empirical results have highlighted how this further feature is not so relevant for a good classification performance. Finally the best accuracy achieved has been 98% for the destination aggregation case and 94% for the source aggregation case, showing a very satisfying result evaluation for the implemented system with a low error ratio. The presence of some False Positive rates is due to a small imperfection in the pre-filtering operation, which leaves some applicative traffic in the background class, especially some tunneled HTTP connections over a non-standard port.

Acknowledgements

Foremost, I would like to thank Prof. Dario Rossi from TELECOM ParisTech, who gave me the chance to develop this thesis within the INFRES department and who shared with me a lot of his expertise and research insight, and prof. Marco Mellia for his great willingness and precious guidelines. I would also like to express my gratitude to Ing. Paolo Veglia and Ing. Silvio Valenti, whose thoughtful advises and technical support allowed me to carry out my work focusing on its "core" and limiting boundary value problems, and to Ing. Alessandro Finamore who trasmitted me his experience and helped me thanks to his past works. Finally, thanks to my closest friends and to anyone else helped me in these months also with just a suggestion or an encouraging: all these supports have been actually decisive and I won't forget about them.

Contents

List of Tables

List of Figures

Introduction

The Internet has never been so dynamic and in continuous evolution as nowadays: users are always changing their habits and tastes and, as a consequence, network traffic is deeply influenced. We are then facing the effects of a huge increase of transmitted data volume and of a quick transformation in the variety of applicative services carried by IP. In fact the introduction of voice, video and other real-time applications has changed the way the Internet is used. Moreover these previous traffics need a strict Quality of Service since their fruition is strongly time and delay dependent. This has triggered the need of monitoring continuously the network to find out which is the average usage of each application in order to allocate the necessary resources and to apply class policies throughout the IP network. Traffic classification meets these specific needs. In the past this type of analysis was carried out by means of simple port-based mechanisms, but currently this is no more feasible since peer-to-peer traffic, which takes up a considerable portion of the total Internet traffic, doesn't use a prefixed port but this is arranged between hosts. Obliged to migrate to other classification mechanisms, packets inspectors were then exploited and they were based on the utilization of a finite state machine to describe the protocol dynamic status. But this system turned out soon to be not so easy to implement and it can't cope with proprietary protocols since it suppose to know their exact dynamic. A statistical approach can avoid this problem thanks to an active learning algorithm which let the system understand the identification features of each application and build a classification model regardless of the knowledge of it. In this thesis work I show a classification method for TCP traffic based on this approach and in particular the decision mechanism is derived from the analysis of the Pearson χ^2 parameter, computed over groups of bit extracted from the transport layer payload. This has revealed to be an excellent decisive factor if computed over a sufficiently large number of packets, describing the variability of the single bits analyzed which can be used to determine with a good probability the relevant protocol class. In fact thanks to it we are able to find out if a group of bit is used as a fixed value, a random one, a counter or a mixed number, that's to say composed of some fixed bits and some random bits. In this way we can obtain a vector of χ^2 values, one for each group of bits which I consider (they are here called chunks). In order to put this concept into practice I implemented a specific module for Tstat software which, starting from an input trace, is able to extracts chunks of bits from the TCP packet payloads and compute over them vectors of statistics with a fixed periodicity. The module permits to define the number of packets to process for any flow analyzed in order to support the idea that the classification results for any applicative protocols is strongly related to the number of packets of each flows which the statistics are computed over. These vectors of values have been then used to train a Support Vector Machine classifier which is able to handle data as points mapped in a multi dimensional space where clustering is easier to solve. From another point of view this is the same as considering the space subdivided in areas associated to a single protocol: the idea is to assign a point to the most likely class according to the area it falls into. If done in a multidimensional space the areas separation process is mathematically (and therefore also computationally) easier. The decisional mechanism has been committed to the already well tested LibSVM library which provides some tools to train, test and make some optimization in the classification process. The goodness of this model is rendered in terms of True Positive, False Positive and False Negative rates computed over the resulting testbed classification.

Chapter 1

Theoretical constructs

1.1 The χ^2 fitting test

The acronym SPI will be used here to indicate Signature Packet Inspection which is the mechanism linking each packet to a signature, that's to say a vector of values computed over portions of the packet's payload. Signatures are computed by means of the χ^2 *goodness of fit test*.

The Pearson's χ^2 test is a well known statistical tool to verify that the frequency distribution of samples in a given test is consistent with a particular theoretical distribution. Its name comes from Karl Pearson (1857-1936), mathematician who established the discipline of mathematical statistics and in particular who firstly investigated this parameter.

We suppose to have a random variable X assuming n different fixed values $\{x_1, x_2, \ldots x_n\}$ and following an unknown distribution which has to be compared to another random variable Y having probability distribution $P(Y = x_i) = p_i$ where $i \in [1, n]$. After N tests we can obtain a result frequency O_i for each of the n values X can assume. The χ^2 statistic is then calculated by finding the difference between each observed and theoretical frequency for each possible outcome, squaring them, dividing each by the theoretical frequency, and taking the sum of the results:

$$\begin{aligned} \widehat{\chi}^2_{n-1} &= \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \\ &= \sum_{i=1}^{n} \frac{(O_i - N \cdot p_i)^2}{N \cdot p_i} \qquad (1.1) \end{aligned}$$

where

1. $\widehat{\chi}^2_{n-1}$ stands for the estimated value with $n-1$ degrees of freedom;
2. O_i is the frequency for the i^{th} value of X;
3. E_i‘ is the expected frequency for the i^{th} value of Y;
4. n is the number of possible outcoming values.

It's plain to see how the chi-square is a gap indicator between O_i empiric and E_i theoretic values, so the smallest is the value, the biggest is the similarity of the random processes X and Y. We can then define a significance level α as a decisional test to verify if X follows the Y distribution and so we can build the following hypothesis test:

$$\begin{cases} H_0 \;:\; \chi^2_{n-1,\alpha} \leq \widehat{\chi}^2_{n-1} & (X \text{ follows the distributional law of } Y) \\ H_1 \;:\; \chi^2_{n-1,\alpha} > \widehat{\chi}^2_{n-1} & (X \text{ doesn't follow the distributional law of } Y) \end{cases}$$

where

1. H_0 stands for the *null hypothesis*, or the default hypothesis according to with the two distributions are equal;
2. H_1 stands for the *alternate hypothesis*, or the hypothesis supported when H_0 is rejected;
3. α defines the probability to reject erroneously the null hypothesis (*I type error*);
4. $n-1$ are the degrees of freedom: the bigger this value is, the more simmetric is the distribution.

So $\chi^2_{n-1,\alpha}$ can be considered as a threshold which must be surpassed by the goodness of fit test to be able to state that X follows Y distribution. A graphical representation is given in figure 1.1: given n and α, if the $\chi^2_{n-1,\alpha}$ falls in the coloured area limited by the α treshold then the test result is negative and the null-hypothesys is refused.

A possible use of the χ^2 test can be in order to verify the uniformity of the output sequence of a random number generator and its randomness, meaning the ability to

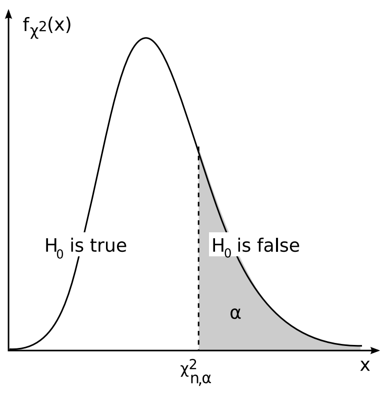


Figure 1.1. Representation of the χ^2 test behaviour.

create sequences of not correlated numbers. Intuitively it would be expected that the goodness of a random generator is related to low χ^2 values because an optimal generator is expected to output numbers with frequencies O_i which should tend to converge to the theorical frequencies E_i: this should result in an approximately null χ^2_{n-1} value. But in this way we're confusing the randomness concept with the uniformity one: if a dice throw always results in the same pattern {1, 2, 3, 1, 2, 3, etc.} it is clear that the samples are uniformally distributed but the sequence has no randomness since it repeats itself periodically. So if a number generator creates an always null χ^2_{n-1} sequence this indicates a deterministic and non-random behaviour. An example of this behaviour is shown in figure 1.2 where the red line stands for the χ^2 test of a true random generator while the green one is associated to the outcome of a counter. With the raise of the number of generated samples the counter's χ^2 converge to a zero value while the random sequence shows an always changing χ^2 value.

So, thus far, it has been shown the ability of the χ^2 parameter to successfully identify a true random sequence from a counter output. However, it would be more useful if it could distinguish also among hybrid cases between the two previous classes, that's to say, it would be interesting to be able to classify a group of bits according to how many of them are random, how many deterministic and how many are counters: in this way it is possible to obtain a restricted number of mixed classes. Some analytical studies results follow, always using the uniform probability

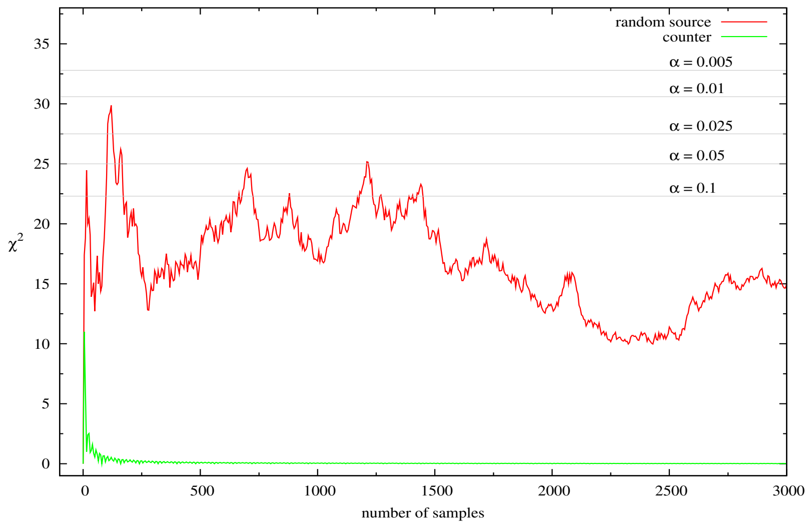


Figure 1.2. Representation of the χ^2 test behaviour.

distribution as the reference for theoretic frequencies.

1.1.1 χ^2 computed over a deterministic group of bits

A group of b bits is said to be deterministic if after any new extraction its value remains constant. In terms of χ^2 computation this means that there is only an i value such that $O_i \neq 0$ while for any other of the remaining $2^b - 1$ values we obtain $O_i = 0$. So *formula* 1.1 can be updated as:

$$\begin{aligned}
\chi^2 &= \sum_{i=1}^{2^b} \frac{(O_i - E_i)^2}{E_i} \\
&= \frac{(2^b - 1)E + (N - E)^2}{E} \\
&= \frac{(2^b - 1)E + (2^b - 1)^2 E^2}{E} \\
&= (2^b - 1)N \qquad (1.2)
\end{aligned}$$

It's evident that in this case the χ^2 varies linearly with the number of extractions by an exponential angular coefficient.

1.1.2 χ^2 computed over a mixed group of bits

We define a mixed-k group of b bits as a set where k out of b bits remain constant independently from the number of extracted samples. For instance if an extraction trial is performed by only considering even (or odds) numbers we obtain groups of bits where the least significant bit is always set as 1 (or 0 for the odd case): the result is a mixed-1 group, where $b-1$ bits are random while the remaining one is deterministic. Also in this case we can rewrite the *formula* 1.1 by classifying the extracted samples as belonging to the following sets:

- $\mathbb{K}_1 = \{$ numeric values not containing the k fixed bits $\}$ with $\|\mathbb{K}_1\| = 2^b - 2^b - k$;
- $\mathbb{K}_2 = \{$ numeric values containg the k fixed bits $\}$ with $\|\mathbb{K}_2\| = 2^b - k$.

So for every samples belonging to $\mathbb{K}_1$, since they are not mixed groups, $O_i = 0$ holds, while samples from $\mathbb{K}_2$ have always $O_i \neq 0$. This property can be exploited and a new χ^2 computation can be carried out separately for the sets of values as it follows:

$$\begin{aligned}
\chi^2_{\mathbb{K}_1} &= \sum_{i\,\in\,\mathbb{K}_1} \frac{(O_i - E_i)^2}{E_i} \\
&= \left(2^b - 2^{b-k}\right) E \\
&= \left(2^b - 2^{b-k}\right) \frac{N}{2^b} \\
&= \left(1 - \frac{1}{2^k}\right) N \\
&= \frac{2^k - 1}{2^k} N
\end{aligned} \tag{1.3}$$

$$\begin{aligned}
\chi^2_{\mathbb{K}_2} &= \sum_{i\,\in\,\mathbb{K}_2} \frac{(O_i - E_i)^2}{E_i} \\
&= \sum_{i\,\in\,\mathbb{K}_2} \frac{(O_i - \frac{N}{2^b})^2}{\frac{N}{2^b}}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i\,\in\,\mathbb{K}_2} \frac{(O_i - \frac{N}{2^b})^2}{\frac{N}{2^{b-k}}} 2^k \\
&= 2^k \sum_{i\,\in\,\mathbb{K}_2} \frac{(O_i - \frac{N}{2^{b-k}})^2 \;-\; \left(\frac{N}{2^{b-k}}\right)^2 \frac{2^{2k}-1}{2^{2k}} \;+\; 2O_i \frac{N}{2^{b-k}} \frac{2^k-1}{2^k}}{\frac{N}{2^{b-k}}} \\
&= 2^k \chi^2_{2^{b-k}-1,\,\alpha} + 2^k \left[-\frac{N}{2^{b-k}} \cdot \frac{2^{2k}-1}{2^{2k}} \cdot 2^{b-k} + 2N \frac{2^k-1}{2^k} \right] \\
&= 2^k \chi^2_{2^{b-k}-1,\,\alpha} + N \left[\frac{-2^{2k}+1+2(2^k-1)2^k}{2^k} \right] \\
&= 2^k \chi^2_{2^{b-k}-1,\,\alpha} + N \frac{2^{2k} - 2 \cdot 2^k + 1}{2^k} \\
&= 2^k \chi^2_{2^{b-k}-1,\,\alpha} + N \frac{(2^k-1)^2}{2^k} \qquad (1.4)
\end{aligned}
$$

Now by merging *formula* 1.3 and *formula* 1.4 it is possible to compute the χ^2 for a generic mixed-k group of bits:

$$
\begin{aligned}
\chi^2 &= \sum_{i=0}^{2^b} \frac{(O_i - E_i)^2}{E_i} \\
&= \underbrace{\sum_{i\,\in\,\mathbb{K}_1} \frac{(O_i - E_i)^2}{E_i}}_{\chi^2_{\mathbb{K}_1}} \;+\; \underbrace{\sum_{i\,\in\,\mathbb{K}_2} \frac{(O_i - E_i)^2}{E_i}}_{\chi^2_{\mathbb{K}_2}} \\
&= \frac{2^k-1}{2^k} N \;+\; 2^k \chi^2_{2^{b-k}-1,\,\alpha} \;+\; N \frac{(2^k-1)^2}{2^k} \\
&= 2^k \chi^2_{2^{b-k}-1,\,\alpha} + N(2^k - 1) \qquad (1.5)
\end{aligned}
$$

The obtained formula holds true only for $0 \leq k < b$ since it doesn't take into account the deterministic case. Two separated components can be detected: the first addend stands for the casual element which depends on the chosen significativity α, the second one is the linear component which grows with the number of extractions and whose angular coefficient varies with k. This means that graphically the χ^2 should approximate a straight line with a k dependent inclination but it should also be subject to a random drift. In order to show the godness of these analytical

results it has been decided to perform a numeric simulation by using a random 4-bits number generator and then by properly manipulating the sequence of bits in order to obtain all possible mixed-k groups. The graphical result is displayed in figure 1.3: in the deterministic case (`MIX4`) we obtain an asymptote without any casual drift while in the other cases the result is a line with an inclination depending on the number of fixed bits.
It's evident how the different groups of bits take up well defined regions of the 2D cartesian space and this property is the key-concept of the classification method developed.

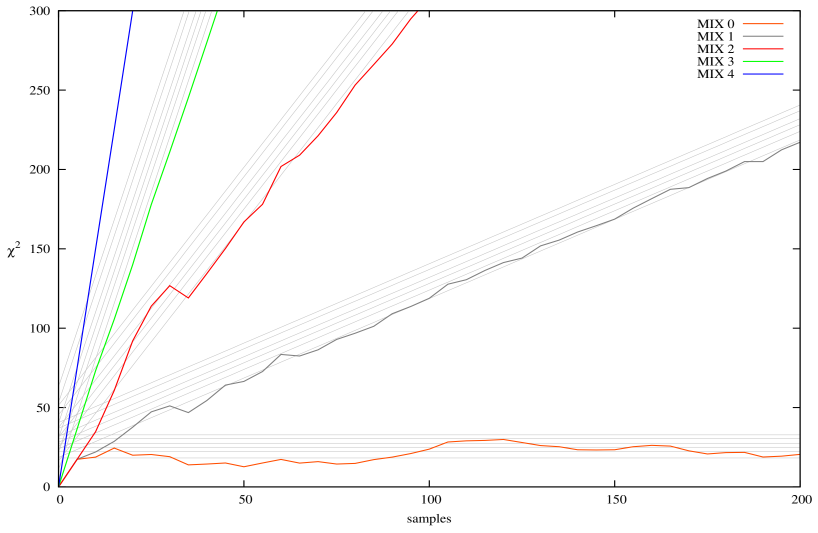


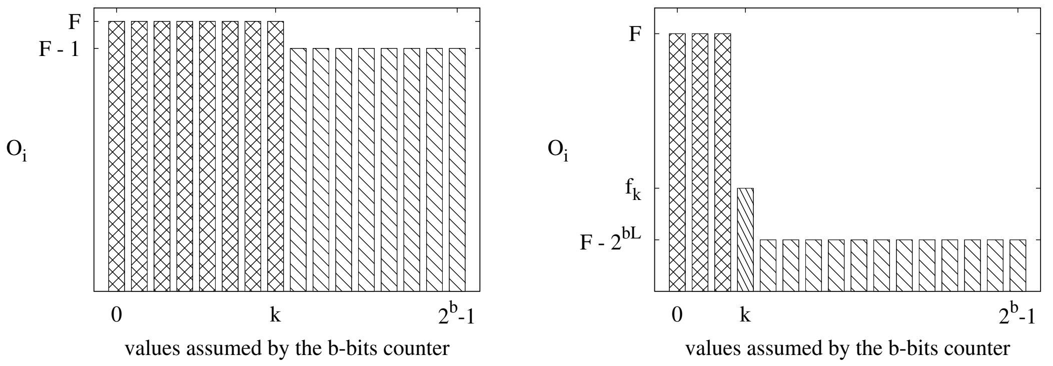
Figure 1.3. χ^2 simulation results for any possible mixed-k 4 bits groups. Lines in grey are the thresholds for a single-queue test by using $\alpha = \{0.005, 0.01, 0.025, 0.05, 0.1, 0.25\}$.

1.1.3 χ^2 computed over a counter group of bits

If a set of integer random numbers is extracted and observed as an unique group composed of b bits and it follows the formula

$$\begin{cases} x_i = & x_0 & (i = 0) \\ x_i = & (x_{i-1} + p) \,\%\, 2^b & (i > 0) \end{cases}$$

where x_i is the i^{th} value extracted, p is the periodicity and b the number of bits, then this group is a counter. In other words for each extraction the group only assumes values with a p periodicity and cyclically with respect to the 2^b different values that the group can assume. Focusing on an unitary period counter at a given time the frequency distribution O_i can be depicted as shown in figure 1.4 where before value k the frequency distribution assumes an F value while after that it assumes value $F-1$. The motivation for the presence of this step is that when k is extracted all the inferior values have already been incrementd and they have reached the F frequency while any other higher value has not yet been incremented and it is at the $F-1$ level.



(a) Frequency of the counter seen in its whole (b) Frequency of a group of bits at L level

Figure 1.4. O_i frequency distribution for a 1-counter

It's possibile to deduce that the χ^2 is symmetrical according to k, particularly $\chi^2_{k=0} = \chi^2_{k=2^b-1}$, $\chi^2_{k=1} = \chi^2_{k=2^b-2}$, ... Furthermore at $\chi^2_{k=2^b-1}$ there should be a maximum point because this is the point where the distribution appears to be the most dissimilar to the uniform one.
These considerations hold true only if the counter is considered in its entirety or, if considering only a subset of all bits, they represent the least significant component of the number. Rather, if the group extracted is broken up in blocks of bits, then the χ^2 behaviour is subject to changes. For example considering a 1byte counter splitted into two 4bits blocks whose the first is called the *Most Significant Group* (*MSG*) and the second the *Least Significant Group* (*LSG*) it is easy to deduce that the LSG's behaviour is coinciding to that of a 1-counter since at each extraction it is incremented with periodicity 1, while the MSG's distribution is characterized by a 16 wide step since it has to wait for 16 extractions (all possible values of the LSG)

before to be incremented. So in this case at the extraction of the value k for all preceding values the frequency distribution assumes an F value, all next values are yet at $F-16$ level while the k value is progressively passing from a $F-16$ level to F. The motivation for the presence of this step is that when k is extracted all the inferior values have already been incrementd and they have reached the F frequency while any other higher value hasn't yet been incremented and it is at the $F-1$ level.
To generalise this concept a new L parameter is here introduced to define the group depth according to the LSG (e.g. in the previous case of a 1byte counter splitted into two 4bits groups, $L=\frac{1}{2}$) and the frequencies' behaviour of a generic group at L-level is depicted in figure 1.4(b). The step width is here related to the depth level while the courrent value k has a progressively changing and unknown frequency f_k which is moving from the inferior $F-2^{bL}$ to the superior F one.
After all the previous considerations it is possible to formalise the presence of the frequency step in the following way:

$$N = kF + 1\cdot f_k + (2^b-k-1)(F-2^{bL}). \tag{1.6}$$

Since f_k is time dependent it should be easier to approximate it by using its inferior and superior limits as following:

- forcing $f_k = F$ (superior limit):

$$\begin{aligned} N &= (k+1)F + (2^b-k-1)(F-2^{bL}) \\ &= 2^bF + 2^{bL}(k+1-2^b) \\ F &= \frac{N-2^{bL}(k+1-2^b)}{2^b} \\ &= E + \frac{2^{bL}}{2^b}(2^b-k-1) \end{aligned} \tag{1.7}$$

- forcing $f_k = F-2^{bL}$ (inferior limit):

$$\begin{aligned} N &= kF + (2^b-k)(F-2^{bL}) \\ &= 2^bF + 2^{bL}(k-2^b) \\ F &= \frac{N-2^{bL}(2^b-k)}{2^b} \\ &= E + \frac{2^bL}{2^b}(2^b-k) \end{aligned} \tag{1.8}$$

In both *formula* 1.7 and *formula* 1.8 $\frac{2^{bL}}{2^b}$ is not simplified because resulting formulas wouldn't be valid for $L = 0$. Now it is possible to develop *formula* 1.1 with the achieved results:

$$\begin{aligned}
\chi^2_{up} &= \sum_{i=0}^{2^b-1} \frac{(O_i - E_i)^2}{E_i} \\
&= (k+1)\frac{(F-E)^2}{E} + (2^b - k - 1)\frac{(F - 2^{bL} - E)^2}{E} \\
&= \frac{k+1}{E}\left[\frac{2^{bL}}{2^b}(2^b - k - 1)\right]^2 + \frac{2^b - k - 1}{E}\left[\frac{2^{bL}}{2^b}(k+1)\right]^2 \\
&= \frac{1}{E}\frac{2^{2bL}}{2^{2b}}\left[(k+1)(2^b - k - 1)^2 + (2^b - k - 1)(k+1)^2\right] \\
&= \frac{1}{E}\frac{2^{2bL}}{2^{2b}}(k+1)(2^b - k - 1)2^b \\
&= \frac{2^{2bL}}{N}(k+1)(2^b - k - 1) \qquad (1.9)
\end{aligned}$$

$$\begin{aligned}
\chi^2_{down} &= \sum_{i=0}^{2^b-1} \frac{(O_i - E_i)^2}{E_i} \\
&= k\frac{(F-E)^2}{E} + (2^b - k)\frac{(F - 2^{bL} - E)^2}{E} \\
&= \frac{k}{E}\left[\frac{2^{bL}}{2^b}(2^b - k)\right]^2 + \frac{2^b - k}{E}\left[\frac{2^{bL}}{2^b}k\right]^2 \\
&= \frac{1}{E}\frac{2^{2bL}}{2^{2b}}\left[k(2^b - k)^2 + (2^b - k)k^2\right] \\
&= \frac{1}{E}\frac{2^{2bL}}{2^{2b}}k(2^b - k)2^b \\
&= \frac{2^{2bL}}{N}k(2^b - k) \qquad (1.10)
\end{aligned}$$

The resulting two formulas are equivalent insofar as the value $(k+1)(2^b - k - 1)$ in *formula* 1.7 represents an unitary horizantal translation towards the left of $k(2^b - k)$ from *formula* 1.10. This impose a parabolic behaviour which, combined to an inversely proportional relation with N, it implies the behaviour shown in figure 1.5

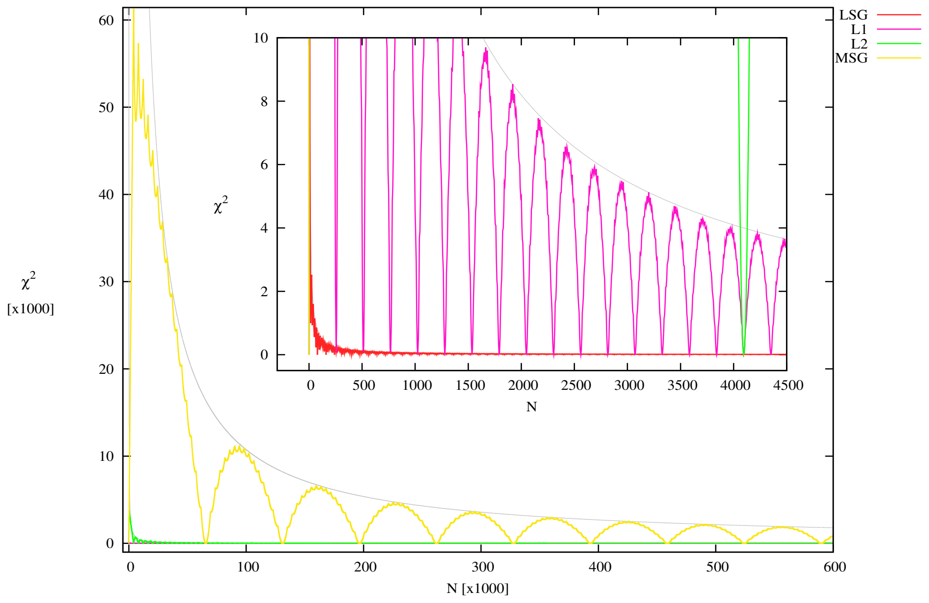


Figure 1.5. Simulation results for a 2byte counter seen as a sequence of four 4bits groups. The internal graph is a zoom into least significant groups and highlights how the general χ^2 behaviour is followed by them too. Lines in grey show the χ^2 envelope described in *formula* 1.10

The parabolic effect is due to the frequency step which, the closer it is to the center of the distribution, the bigger values the χ^2 assumes since the distribution results to be the more different to the uniform one. The maximum points are defined by the hyperbole formula

$$\chi^2_{max} = \frac{2^{2bL}}{N} 2^{2(b-1)} \tag{1.11}$$

which is obtained by *formula* 1.10 by forcing $k = 2^{b-1}$, the value representing the greatest unlikeness between resulting and theoretical distributions. χ^2 periodicity is forced by the relations:

$$\Delta_{0,i} = i \cdot 2^{b(L+1)} \qquad (i = 0, 1, 2 \ldots) \tag{1.12}$$

$$\Delta_{max,i} = \frac{\Delta_{0,1}}{2} + \Delta_{0,i} \qquad (i = 0, 1, 2 \ldots) \tag{1.13}$$

where $\Delta_{0,i}$ e $\Delta_{max,i}$ identify the number of increments needed in order to obtain respectively the i^{th} null value and the i^{th} max value from χ^2. Relationship with N

is related to the fact that the step width is constant, so with the increasing of the number of increments, differences among various O_i frequencies becomes meaningless and the χ^2 value converge to zero. Different counters are identified by the speed at wich their χ^2 converge to zero and formally this is caused by the value 2^{2bL} which implies a minor speed for the greatest levels since their increments depend on those with inferior levels.

1.2 Support Vector Machine classifier

SVM Support Vector Machine is a useful technique for data classification and regression based on the Statistical Learning Theory (*SLT*) recently emerged as a general mathematical framework for estimating dependencies from finite samples. This class of learning system has proved to be more efficient than neural networks in helping avoid the *overfitting* problem.
The statistical learning theory is a mathematical discipline concerning the classification problems modelling and the research of the system's solutions and it was introduced in the middle of the '90s by Vladimir Vapnik who worked out the basis concepts in [11] and [12] and later developed thanks to the interest of the whole scientific commettee. In fact this new statistical science has been exploited in a large number of application fields such as biology, human behaviour's studies, generical objects classification, genetics and so on, completing the long list with the traffic networks field.

1.2.1 Statistical learning theory issues

The easiest learning problem is a binary classification where an object belongs to only one out of two classes. Supposing to have a pre-classified set of objects defined as

$$\mathcal{X} \times \mathcal{Y} = \{x_1, x_2, \ \dots \ , x_m\} \times \{+1 \ ; \ -1\}$$

where $\mathcal{X}$ is a finite set of objects while $\mathcal{Y}$ is a set of classes (also named labels). The problem's solution research consists of identifying the $f(x)$ function predicting the correct class of sample x after an exploitation of the existing relationships between sets $\mathcal{X}$ and $\mathcal{Y}$. This means that beginning from the objects analysis all existing relationships among data should be found out and later expressed in terms of a *likelihood function*, that's to say a function able to identify any object's class because it is "similar" to particular elements belonging to that class. For example, if considering only samples belonging to a generical vector space, a possible likelihood function can be the following scalar product:

$$likelihood = k\,(x,\,x') = <x,\,x'>$$

This example can be generalized to any cases where the sample space does not contain the scalar product operation, and in this situation a mapping function

$$\begin{aligned} \Phi : \mathcal{X} &\mapsto \mathcal{H} \\ x &\mapsto \mathbf{x} := \Phi(x) \end{aligned} \tag{1.14}$$

is introduced which transfers x samples to a new space $\mathcal{H}$ called *features space* where the scalar product is defined. Putting all together it can be obtained

$$\begin{aligned} likelihood &= k\,(x,\,x') \\ &= k\,(\Phi(x),\,\Phi(x')) \\ &= <\mathbf{x},\,\mathbf{x}'> \end{aligned}$$

where $\mathbf{x}$ and $\mathbf{x}'$ represent the two vectors in the features space associated to samples x and x'. In this way it is possible to change the mapping function in order to perform different analysis according to the problem typology leading to flexibility in data elaboration.
An example is shown in figure 1.6 where the samples space is contained in $\mathbb{R}^2$ and the two classes are separable by an ellipse. How shown in [15], setting up the mapping function as

$$\Phi(x) \;=\; ([x]_1^2,\,[x]_2^2,\,\sqrt{2}[x]_1[x]_2)$$

which performs a non-linear transformation $\mathbb{R}^2 \;\rightarrow \mathbb{R}^3$ such that the classification function corresponds to an hyperplane in the features space.

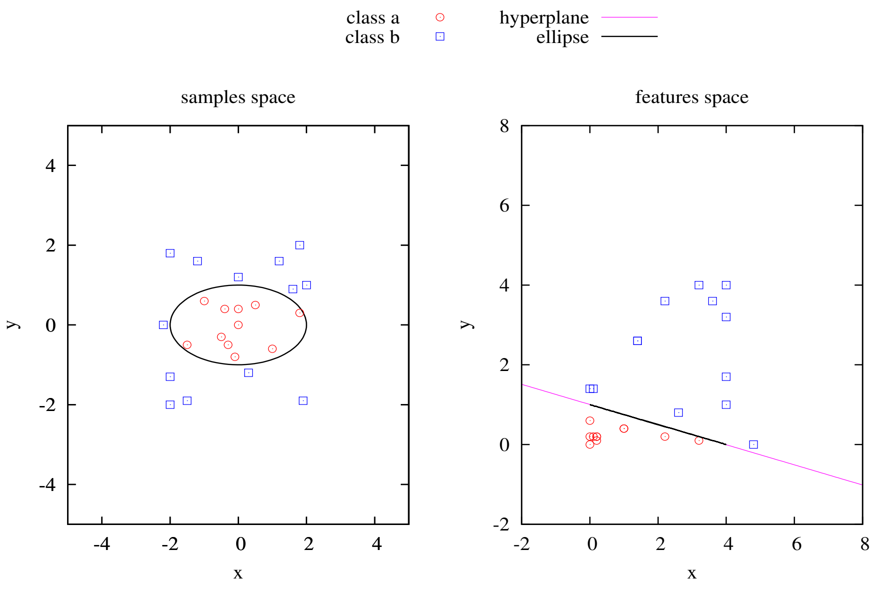


Figure 1.6. Mapping example: the sample space is contained in $\mathbb{R}^2$ and objects are separated by means of the ellipse $\frac{x^2}{2} + y^2 = 1$. Using the mapping function $\Phi(x) = ([x]_1^2, [x]_2^2, \sqrt{2}[x]_1[x]_2)$ the two classes objects can be separated by an hyperplane contained in the features space whose only the XY plane is here represented.

The pre-classified set of objects $\mathcal{X} \times \mathcal{Y}$ is named *training set* because it is at the base of the learning process permitting to distinguish every samples' class. In order to select an efficient classification function the training set must be representative of all classes and have a finite size, furthermore it is evident that training set dimensioning is outstanding, but having a very detailled set does not lead necessarily to an improve in results. An example can help to find out this concept: if a naturalist having to identify plants classifies in a too detailled fashon, he could have problems in recognising a tree having more branches than normal or, if he's not that detailled, he could not recognize a tree without leaves because he only knows trees with green ones.

Supposing to have a representative training set, it is possible to formalize the solution of this binary problem. Defining c_+ and c_- as follows

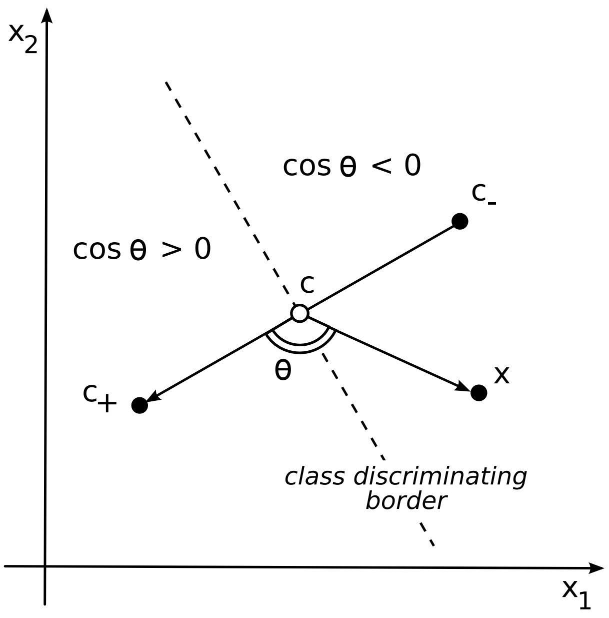


Figure 1.7. Graphical representation of a binary classification problem with the vector product as likelihood function

$$\mathbf{c}_+ = \frac{1}{m_+} \sum_{i \,|\, x_i \,\in\, \mathcal{X}^+} \mathbf{x}_i \tag{1.15}$$

$$\mathbf{c}_+ = \frac{1}{m_-} \sum_{i \,|\, x_i \,\in\, \mathcal{X}^-} \mathbf{x}_i \tag{1.16}$$

as the cendroids of the samples belonging to the two separated classes and given a generic object $\mathbf{x}$, it must be performed a classification based on the minimum distance, that's to say that $\mathbf{x}$ is assigned to the closer class. As depicted in figure 1.7 the discrimination is carried out by the medium point between centroids so that if $\mathbf{x}$ is at its right then it is labelled as belonging to class "-", otherwise it is associated to class "+". In a more formal way the classification function can be written down as

$$\begin{aligned} f(x) &= sgn\Big(<\mathbf{x} - \mathbf{c},\, \mathbf{c}_+ - \mathbf{c}_+> \Big) \\ &= sgn\Big(<\mathbf{x} - \frac{(\mathbf{c}_+ + \mathbf{c}_+)}{2},\, \mathbf{c}_+ - \mathbf{c}_+> \Big) \end{aligned}$$

$$\begin{aligned} &= sgn\Big(<\mathbf{x},\, \mathbf{c}_+> \,-\, <\mathbf{x},\, \mathbf{c}_+> \,+ \frac{1}{2}(||\mathbf{c}_+||^2 - ||\mathbf{c}_+||^2)\Big) \\ &= sgn\Big(<\mathbf{x},\, \mathbf{c}_+> \,-\, <\mathbf{x},\, \mathbf{c}_+> \,+ b\Big) \qquad (1.17) \end{aligned}$$

In the last row an offset has been considered by introducing the b term related to the centroids' distance from the reference system origin. Now it is possible to merge *formula* 1.16 and *formula* 1.15 with *formula* 1.17 in order to obtain the expression defining the classification function accordingly to the training set.

$$\begin{aligned} f(x) &= sgn\Big(\frac{1}{m_+} \sum_{i\,|\,x_i \in \mathcal{X}^+} <\mathbf{x},\, \mathbf{x}_i> - \frac{1}{m_-} \sum_{i\,|\,x_i \in \mathcal{X}^-} <\mathbf{x},\, \mathbf{x}_i> + b_k \Big) \\ &= sgn\Big(\frac{1}{m_+} \sum_{i\,|\,x_i \in \mathcal{X}^+} k\,(x,\, x_i) \,-\, \frac{1}{m_-} \sum_{i\,|\,x_i \in \mathcal{X}^-} k\,(x,\, x_i) \,+\, b_k \Big) \qquad (1.18) \end{aligned}$$

where

$$\begin{aligned} b_k &= \frac{1}{2m_+^2} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^+} <\mathbf{x}_i,\, \mathbf{x}_j> - \frac{1}{2m_-^2} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^-} <\mathbf{x}_i,\, \mathbf{x}_j> \\ &= \frac{1}{2m_+^2} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^+} k\,(x_i,\, x_j) - \frac{1}{2m_-^2} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^-} k\,(x_i,\, x_j) \end{aligned}$$

This new formulation highlights that the previous solution contains a probabilistic element: if $k(\cdot)$ function, expressing the likelihood between two samples, is considered as the joint probability density

$$\int_{\mathcal{X}} k\,(x,\, x')\, dx = 1 \qquad \forall x' \in \mathcal{X}$$

then *formula* 1.17 can be reformulated accordingly to Bayes probability:

$$f(x) = sgn\Big(\underbrace{\frac{1}{m_+} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^+} k\,(x_i,\, x_j)}_{p(x\,|\,x_+)} - \underbrace{\frac{1}{m_-} \sum_{i,\,j\,|\,x_i,\,x_j \in \mathcal{X}^-} k\,(x_i,\, x_j)}_{p(x\,|\,x_-)} + b_k \Big) \qquad (1.19)$$

The above expression now clearly presents a functioning based to the displacement between probabilities to belong to each class: the object is associated to the class with the higher probability, and this is another way to define the minimum distance mechanism. If reformulating *formula* 1.19

$$
\begin{aligned}
y &= sgn\left(\frac{1}{m_+}\sum_{i\,|\,x_i\in\mathcal{X}^+} k\,(x,\,x_i) \;-\; \frac{1}{m_-}\sum_{i\,|\,x_i\in\mathcal{X}^-} k\,(x,\,x_i) +\; b_\kappa\right) \\
&= sgn\left(\sum_{i=1}^{m}\alpha_i\, k\,(x,\,x_i) + b_k\right) && (1.20) \\
&= sgn\,(<\mathbf{w},\,\mathbf{x}> \;+ b) && (1.21)
\end{aligned}
$$

it results more evident how the feature space is divided by a discriminating hyperplane defined by the vector **w**, orthogonal to the hyperplane and whose components are represented by α_i, by the likelihood function $k(\cdot)$ and by b, again representing the hyperplane offset with respect to the origin.

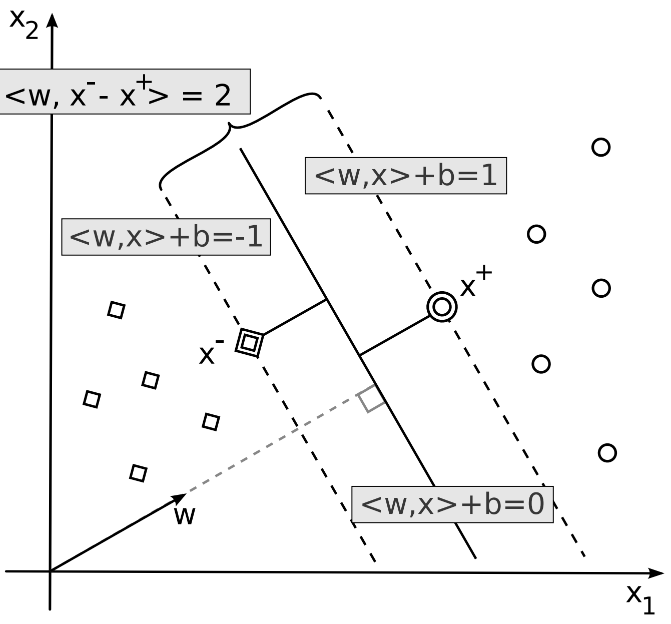


Figure 1.8. Graphical representation of a separating hyperplane defining two classification zones in the features space

1.2.2 Support Vector Machine - The separable Case

In the previous subsection the solution of the classification problem has been presented in terms of discriminating hyperplanes identified by **w** and b. But in order to obtain an efficient classification the optimal solution should be selected among all possible hyperplanes, and this is represented by the one maximizing the space that separates the clouds of objects belonging to different classes, also called the *margin*.

As said before $\mathbf{w}$ is the vector normal to the hyperplane, $\frac{b}{||\mathbf{w}||}$ is the perpendicular distance from the hyperplane to the origin, and $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$.Let d_+ (or d_-) be the shortest distance from the separating hyperplane to the closest positive (or negative) example. Define the margin of a separating hyperplane to be $d_+ + d_-$, now the support vector algorithm looks for the separating hyperplane with largest margin. This can be formulated as follows: suppose that all the training data satisfy the following constraints:

$$< \mathbf{w}, \mathbf{x}_i > +b \geq 1 \quad \forall i \mid x_i \in \mathcal{X}^+ \tag{1.22}$$

$$< \mathbf{w}, \mathbf{x}_i > +b \leq -1 \; \forall i \mid x_i \in \mathcal{X}^- \tag{1.23}$$

These can be combined into one set of inequalities:

$$y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 0 \tag{1.24}$$

Now consider the points for which the *formula* 1.22 holds: these points lie on the hyperplane $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ with normal $\mathbf{w}$ and perpendicular distance from the origin $\frac{1-b}{||\mathbf{w}||}$. Similarly, the points for which the equality in *formula* 1.23 holds lie on the hyperplane $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, with normal vector equal to $\mathbf{w}$, and perpendicular distance from the origin $\frac{-1-b}{||\mathbf{w}||}$. Hence $d_+ = d_- = \frac{1}{\mathbf{w}}$ and the margin is simply $\frac{2}{\mathbf{w}}$. Note that H_1 and H_2 are parallel (they have the same normal vector) and that no training points fall between them. Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $||\mathbf{w}||^2$, subject to *formula* 1.24.

So, we can formalize the optimization problem whose solution can be found out after solving the following well known minimization problem:

$$\begin{cases} \min\limits_{\mathbf{w},\, b} \quad \tau(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \\ y_i(< \mathbf{w}, \mathbf{x}_i > \; + b) \geq 1 \quad i \; \in \; [0,m] \end{cases} \tag{1.25}$$

Minimize $||\mathbf{w}||$ is exactly equivalent to maximise the margin because, as shown before, this represents the inverse of the margin. The choise of minimizing the square of the norm is related to practical issues, because in this way the problem becomes of quadratic type and there exists some efficient algorithms to solve the problem. The constraint forces that points are at a unitary minimum distance from the hyperplane while it can't be set to 0 since it would imply that there could be objects placed on the separation border and than classification would be ambiguous. The binds are multiplied for $y_i \in \{+1;\ -1\}$ representing the class label in order to define the orientation in the hyperplane and this is equivalent to the constraints already expressed in *formula* 1.22 and *formula* 1.23: in other words it has been assumed the existence of a function resulting positive values if the point belongs to class +1 or negative ones if it is associated to class −1.

As it is now possible to guess, since the classification is done by hyperplanes placed in the space so that they maximize the distance between border objects, not all the points are engaged in the solution research because only those in the clouds peripheries are useful involved in margins computation. These objects are called *support vector* since they "bear" the solution and if removed the solution would change. Notice that there is a Lagrange multiplier α_i for every training point. In the solution, support vectors always have $\alpha_i > 0$ and lie on one of the hyperplanes H_1, H_2 . All other training points have $\alpha_i = 0$ and lie either on H_1 or H_2 (such that the equality in *formula* 1.24 holds), or on that side of H_1 or H_2 such that the strict inequality in *formula* 1.24 holds. For these classification machines, the support vectors are the most critical elements of the training set: they lie closest to the decision boundary and if all other training points were removed (or moved around, but so as not to cross H_1 or H_2), and training was repeated, the same separating hyperplane would be found.

Now it is better to switch to a Lagrangian formulation of the problem because the constraints seen till now will be replaced by constraints related to the Lagrange multipliers, which will be much easier to handle. So the minimization problem can be analitically solved with the solution

$$f(x) = sgn\Big(\sum_{i=1}^{m} y_i\alpha_i < \mathbf{x}, \mathbf{x}_i > +b\Big) \tag{1.26}$$

$$b = y_j - \sum_{i=1}^{m} y_i\,\alpha_i\ < \mathbf{x}_i, \mathbf{x}_j > \quad \forall j\ \in\ [1,m] \tag{1.27}$$

which can be formulated in terms of training set as

$$f(x) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b\right) \tag{1.28}$$

$$b = y_j - \sum_{i=1}^{m} y_i\, \alpha_i\, k(x_i, x_j) \quad \forall j \in [1,m] \tag{1.29}$$

α_i parameters are the lagrangian coefficients and they highlight how the solution is generated by a good linear combination of vectors in the features space. It can be also shown in formal way that only for the support vector $\alpha_i \neq 0$. After this, the likelihood function $k(\cdot)$ allows to map again the solution to the samples space. This transformation is performed through the *kernel function* that's exactly what it has been called till now likelihood function. In fact, if the correct likelihood function is known, working in the samples space can ease computation, avoiding difficult vectorial elaborations due to the fact that the problem is expressed in terms of the training set, whose size is decisive. In particular, if in the samples space the training set can be associated to a matrix $m \times d$, mapping the problem in the features space computations are related to a matrix $m \times D$, with $D \geq d$. Then it should be identified a function allowing to cope with vector products without having really to use matrix computations, being this operation the core (or kernel) of the whole classification solving mechanism. Exhaustive researches have found out several kernel functions, the most used shown in table 1.1.

Linear	$k(x, x') = (<x, x'>)$	
Polynomial	$k(x, x') = (\gamma <x, x'> +c)^d$	$d \in \mathbb{R}$, $c > 0$ $\gamma > 0$
RBF Gaussian	$k(x, x') = \exp^{-\gamma\|\|x-x'\|\|^2}$	$\gamma > 0$
Sigmoid	$k(x, x') = tanh(\gamma <x, x'> +\theta)$	$\gamma > 0, \theta < 0$
RBF B_n-spline	$k(x, x') = B_{2p+1}(\|\|x - x'\|\|)$	$I_{[-\frac{1}{2};\, \frac{1}{2}]}(a) = 1$ se $-\frac{1}{2} \leq a \leq \frac{1}{2}$

Table 1.1: List of the most used kernel functions for SVM classification

1.2.3 Support Vector Machine - The Non-Separable Case

The above algorithm for the research of the classificating hyperplane for separable data, when applied to non-separable data, will find no feasible solution: this will be pointed out by the objective function (i.e. the dual Lagrangian) growing arbitrarily large. In order to extend the mechanism also to this case problem an approach is to relax the constraints represented by *formula* 1.22 and *formula* 1.23, but only when necessary, that is, we would like to introduce a further cost (i.e. an increase in the primal objective function) for doing so. This can be done by introducing positive slack variables ξ_i , $i = 1, \ldots ,l$ in the constraints (as described in [14]) , which then become:

$$< \mathbf{w}, \mathbf{x}_i > +b \geq 1 - \xi_i \quad \forall i \mid x_i \in \mathcal{X}^+ \tag{1.30}$$

$$< \mathbf{w}, \mathbf{x}_i > +b \leq -1 + \xi_i \; \forall i \mid x_i \in \mathcal{X}^- \tag{1.31}$$

In this way single points are allowed to be placed in incorrect zones at a cost of an error weight (a *soft margin* is here involved), and the minimization problem can be rewritten as follows:

$$\begin{cases} \min\limits_{\mathbf{w},\, b,\, \xi_i} \quad \tau(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \; + \; C\sum\limits_{i=1}^{m} \xi_i \quad \xi_i > 0 \\ y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 1 - \xi_i \qquad\qquad\qquad i \; \in \; [0,m] \end{cases} \tag{1.32}$$

In the objective function a $C > 0$ parameter is introduced to define the weight of ξ_i error: a larger C corresponds to assigning a higher penalty to errors and to increase endurance against classification noise. For an error to occur, the ξ_i must be greater than the unity because, as it can be seen in the system above, if $0 \leq \xi_i < 1$ the point is correctly classified. Furthermore, the presence of the error affects also the b parameter related to the distance from the origin allowing the hyperplane to move towards the borders in order to minimize the presence of errors.
The C parameter has to be manually chosen but a simple way to set it up is to exploit a process of cross-validation consisting of a series of classification trials modifing the training set in order to find out which is the best C parameter.

1.2.4 Multiclass problems

Till now only the binary classification case has been discussed, while real life situations are related to multiclass problems studies so that it is very important to define this kind of case. Generally a multiclass problem can be expressed as a series of binary subproblems whose predictions can be combined to formulate the final prediction.

There are some different methods to perform this decomposition, for example by using the *one-against-all* approach consisting of sequential repetitions of the classification once for each of the M classes separating the current class from the aggregate of all the others. Then the M classifiers are combined to make the final decision, the classifier which generates the highest value from its decision function is selected as the winner and the corresponding class label is assigned. In other words the outputs of the decision function are employed as the only index to indicate how strong a sample belongs to the class

$$\arg\max_j \quad \sum_{i=1}^{m} y_i \alpha_i^j k\left(x, x_i\right) + b^j \quad j \in [1,M],\, y_i = +1 \Leftrightarrow x_i \in \mathcal{X}^j \tag{1.33}$$

where $\mathcal{X}^j$ is the samples set of each class. This is equivalent to choosing the hyperplane positioning which performs the best classification defining the region more suitable to contain a point.

Another possible approach is the so called *one-against-one* strategy also known as “pairwise coupling”, “all pairs” or “round robin”, which consists in constructing one SVM for each pair of classes. Thus, for a problem with M classes all the possible combinations are

$$\binom{M}{2} = \frac{M(M-1)}{2}$$

and that’s exactly the number of SVMs trained to distinguish the samples of one class from the samples of another class. The solution is chosen with a votation based system, chosing the class that solves the greater number of subproblems. Usually, classification of an unknown pattern is done according to the maximum voting, where each SVM votes for one class.

It’s evident that the second method implies an increase in the number of subproblems, but these are of course of smaller size and so not at all expensive at a computational level, improving the elaboration time. Experimental results of the comparison between the two methods are available in [16]. The common drawback is that both strategies have to solve in an exaustive fashion all the subproblems before arriving to a final solution. Some approaches to the problem have been tested and the most suitable is the so called *DAG - Directed Acyclic Graph*, defining according to what the subproblems have to be solved with at the basis of the idea that some couples of classes are more frequent in the training set than other ones and so they can be thought to be more important. In this way the subproblems are ordered following their relevance and an exaustive research is no more requested.

Chapter 2

Classification workflow

The aim of this work is to build a blind system, a classifier which has no pre-written rules but which is able to identify protocols grounding on traffic-deduced features. In this way it is possible to learn also format informations about proprietary protocols whose dynamic is quite unknown (e.g. Skype). So the scope is that of identifying by means of a χ^2 test the application level protocol carried by any TCP packets payload analyzed. This is possible because, as shown in paragraph 1.1, the analysis of χ^2 vectors related to different bits gives detalled informations about the single bits variability and so, putting all together, it could provide notions about the protocol format. It's not needed to care about specific values of the header fields but just a measure of the randomness is enough to permit an exact identification. In particular, to be sure that applicative protocol informations are considered, attention should be put only on the first bytes of the TCP payload because these usually contain the protocol data header or just a portion of it.

Information regression is committed to the SVM which maps the χ^2 vectors as points into a higher dimensional space. In this way samples belonging to different protocols can be well separated in the cartesian space by an hyper-plane with an high discriminating power. As some efficient implementations of this algorithm are already available, `LIBSVM` library has been adopted, since it provides powerful tools for training and prediction actions and optimization methods such as grid optimization or scaling methods in order to achieve the biggest accuracy with the most limited computational requirement.

As stated before the SVM classifier needs to be instructed in order to create the model and later this model is exploited to perform classification. So the classification model here described can be conceptually split into two parts:

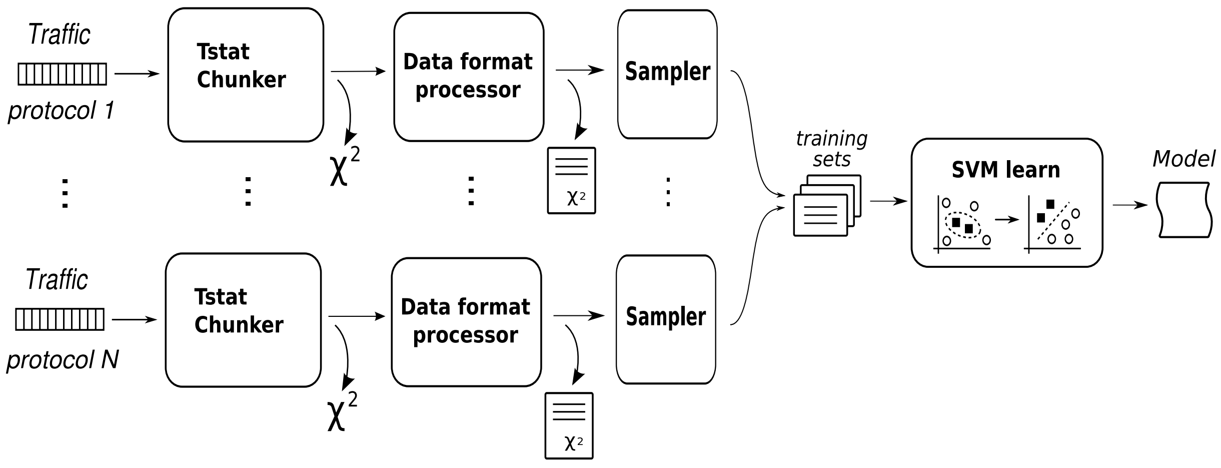
1. the learning process;
2. the prediction process.

The former - scheme 2.1(a) - is intended to produce a model file which will be used during the prediction phase to feed the SVM classifier. The whole traffic trace is previously analized and parted according to the applicative protocol (paragraph 3.1.3 explains how this has been done) and all related χ^2 signatures are independently computed. After this, a random extraction is performed in order to select the samples adopted for the SVM training among all protocols datasets. The method used for that random choice is described in paragraph 3.1.2.
The latter model - scheme 2.1(b) - does traffic prediction: starting from the trace all signatures are computed and treated to be analyzed by the SVM classifier which is able to predict the applicative protocol thanks to the model file worked out during the learning phase.

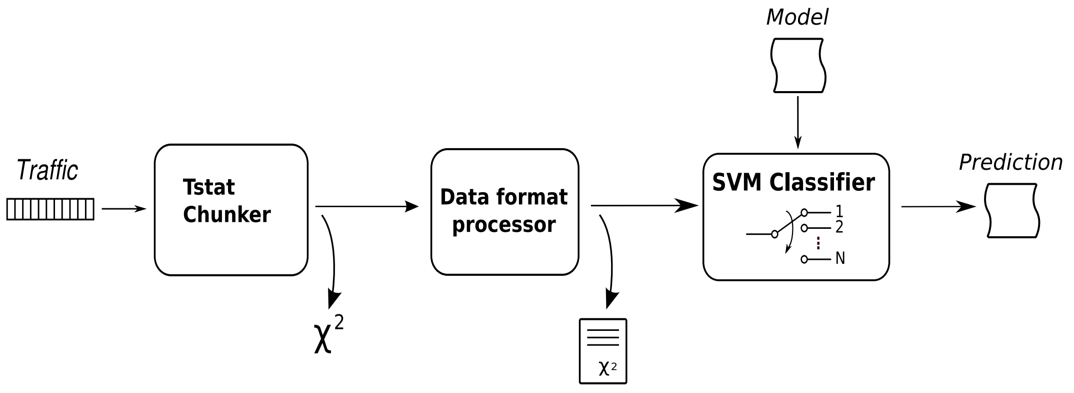
All modules represented in figure 2.1 are described in details in the next paragraphs.

2.1 Classification item

As said before the classification is carried out by means of χ^2 values computed over the TCP packets payload: foremost the number of bytes over which the analysis must be performed is set and the bit dimension of each chunk is chosen. Then during the traffic analysis vectors of every chunk's values are collected and statistics are periodically computed in order to give a variability index with respect to the number of parsed samples. These parameters are then dumped into log files with a given periodicity defined by the probability theory: to obtain a reliable χ^2 value it must be observed a number of samples equal to 6-7 times the cardinality of the possible values which the variable can assume. That means that for a 4bits chunk about a hundred samples are needed, but deterministic and mixed cases have to be considered too, and this decreases for sure the number of samples to wait before computing statistics. It's here evident that this classification system needs to be fed



(a) Learning process model



(b) Prediction process model

Figure 2.1. Classifier generic structure

with a minimum number of packets, but this kind of limitation can be very severe for certain typologies of traffic, so that a kind of aggregation is demanded.

2.2 Transport layer traffic characterization

The object of the analysis is TCP traffic and the classification of its upper layer protocol such as SSH, HTTP, POP3 etc, it is therefore clear that we are coping with a lot of flows which are not so voluminous, and this should represent a problem for statistics computation because of thresholds on the number of packets needed to dump data. For this kind of traffic the only solution is to compute statistics after having aggregated packets by source or destination endpoints, while flow aggregation is not possibile because it is very difficult here to reach a sufficient number of packets for any flow. So with this kind of aggregation we are studying every endpoints' behaviour trying to identify the application they are using. Instead, if considering UDP traffic flows, they are intrinsecally characterized by huge volumes so that endpoint aggregation is not required and can be replaced just with flow aggregation. For further details about UDP traffic flows managements a suggested reading is represented by [9].

2.3 Tstat TCP-chunker

`Tstat` is a free software developed under the terms of the GNU General Public License by the *Telecomunication Networks Group* of Politecnico di Torino and available for Unix environments. Deeply based on `TCPTRACE`, it is a network traffic analysis tool able to trace TCP and UDP flows, to analyze the dynamic of a connection and to compute statistics. It can work both by parsing dump files in an off-line fashion or as a sniffer, so computing statistics and logs in real time: in this work only the first ability has been exploited focusing only on TCP packets analysis.

Since it has a modular structure, any implementation of new features requires the creation of a new component which has to be added to the Tstat plugin interface. In this way for any packet inspected a sequence of auxiliary elaborations is done after the transport layer analysis: once the IP upper layer is found out, it is possible to select all TCP or UDP packets. And that's exactly what it has been done: a new module TCP-chunker has been inserted in order to periodically create vectors of χ^2 computed by aggregating TCP flows by endpoints. A view of what the module does is presented in figure 2.2, showing the statistics generation and dump processes.

The output of this module is a pair of log files containing lists of statistical values related to packets belonging to both flows directions from source to destination and from destination to source. This separation is compulsory since in almost all cases the two communication directions are not symmetric and therefore it is imperative to analyze the clients and servers behaviours independently and to generate two distinct signatures.

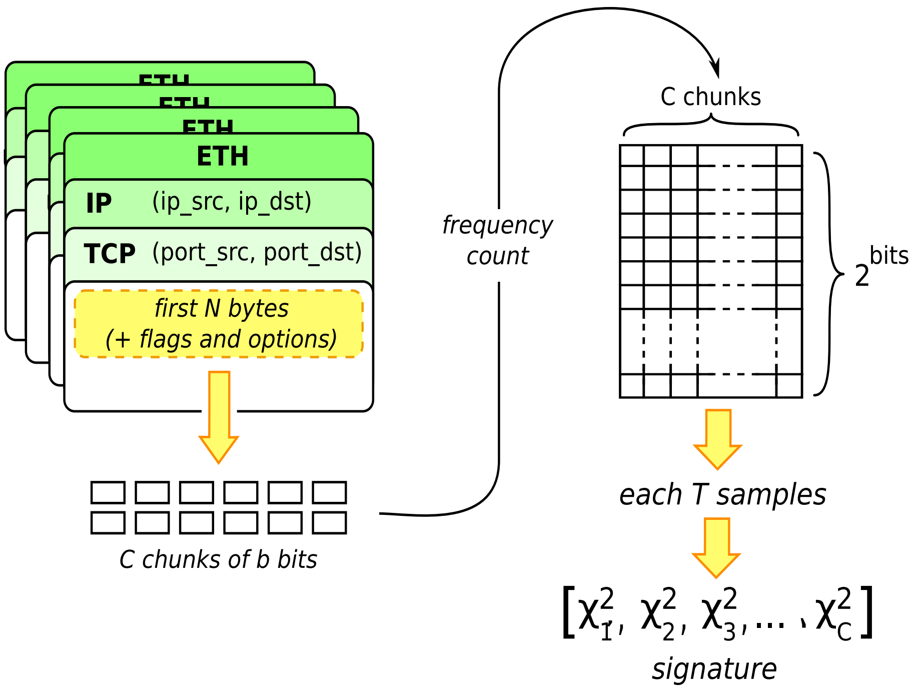
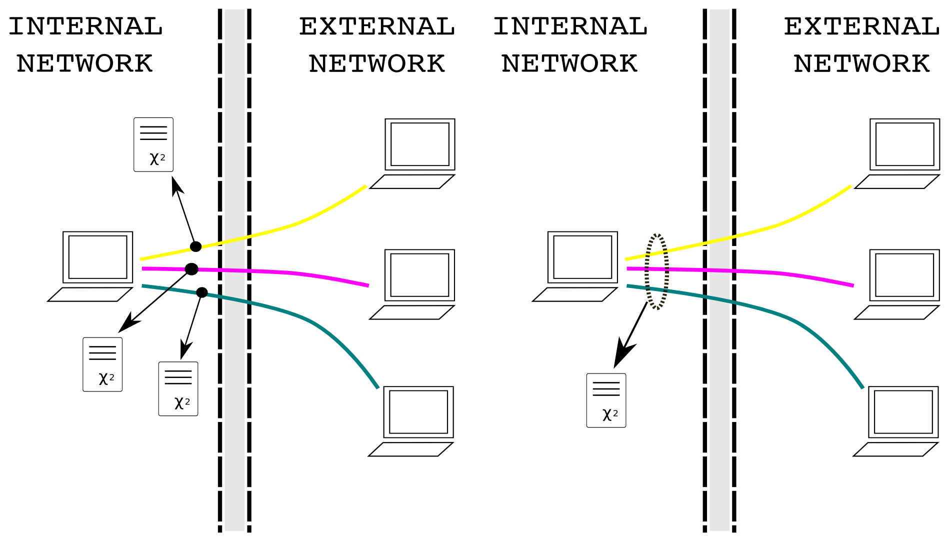


Figure 2.2. Chunking operation and χ^2 generation process

To aggregate by endpoints means that statistics are computed over flows with the same source or destination endpoint. This means that it is possible to acquire informations also about protocols don't generating huge traffic quantities in each single flows, in this case the only limitation is that a minimum number of opened flows is needed in order to obtain the statistics computed. This kind of aggregation is the only possible in this case study since the adopted policy here is to analyze only the first N packets of each flows, considering the idea that they are the ones discriminating most effectively any protocol. In fact in any application service communications always begin with a handshake or a fonctionning information exchange between hosts and after that phase the data transmission is carried out. This part of the conversation is not useful for the purpose of identifying the protocol, so it has been decided to ignore it by considering only the first packets. Because of this policy traffic volume per flow is limited and this forbids us to make a flow aggregation: the endpoint aggregation is the only solution.
Figure 2.3 helps to see through these concepts, showing the difference between flow and endpoint aggregation.



(a) Flow aggregation: a χ^2 signature is computed over every packets of a flow.

(b) Endpoint aggregation: a χ^2 signature is computed over every packets sent to (or received by) an endpoint.

Figure 2.3. Two different ways of performing traffic aggregation.

2.3.1 Tstat basis structures

Any host generating applicative traffic over the TCP protocol is said to be an *endpoint* and it is defined by the pair `ip_address:port` which is enough to univocally identify the application in the network at transport level. Moreover, any packet is associated to a pair `(source endpoint – destination endpoint)` defining the *flow* the packet belongs to and giving informations about the direction of transmission. These relations are established through a group of structures already implemented in Tstat original version and in particular they are organized as shown in figure 2.4.

The main structures are:

- `tcp_pair`: structure containing informations about a single conversation and allowing access to single flows informations according to the transmission direction;
- `addr_pair`: structure including endpoints informations;
- `ttp`: a circular buffer allowing to access any conversation data by means of a

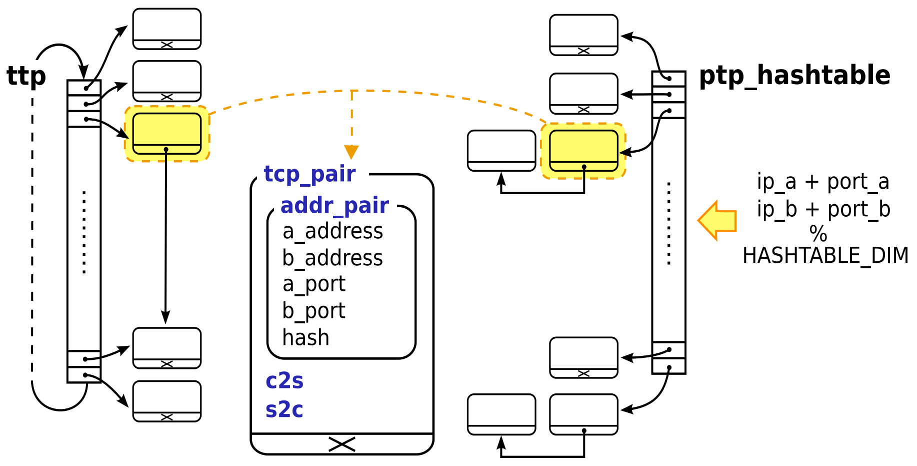


Figure 2.4. Tstat data structure organization managing endpoint, flow and conversation level information and statistics.

linear search;

- `ptp_hashtable`: a list allowing to perform searches based on the conversation-id through a hash which speeds up the seeking process.

In details, `tcp_pair` contains three structures: `addr_pair` which includes the conversation identification code; `c2s` and `s2c` bearing statistics related to each direction. The identification code is written when the conversation starts, that's to say at the first packet received not belonging to any other conversation already stored, and it is represented by the pair of endpoints carrying on the conversation, therefore registering the first direction. This criterion respects the client/server definition, where the client is the host that first perform a request while the server answers and this is also the motivation for the presence of the structures `c2s` and `s2c`.
So when a new packet is analyzed it is firstly associated to a conversation by checking the proper id and later, in order to find the correct direction, a double check is done over the source and destination endpoint to assign it to the correct flow.
To seek the right conversation two methods are available: the `ttp` circular buffer and the `ptp_hashtable`. Since Tstat has to cope with a huge number of flows the linear search over `ttp` is not efficient, so the hash table method is preferred. The hash key is done by the sum of the conversations id in module with the hashtable dimension, granting in this way a fast and easy method to access conversation structures since only a short collision list is to take into account.

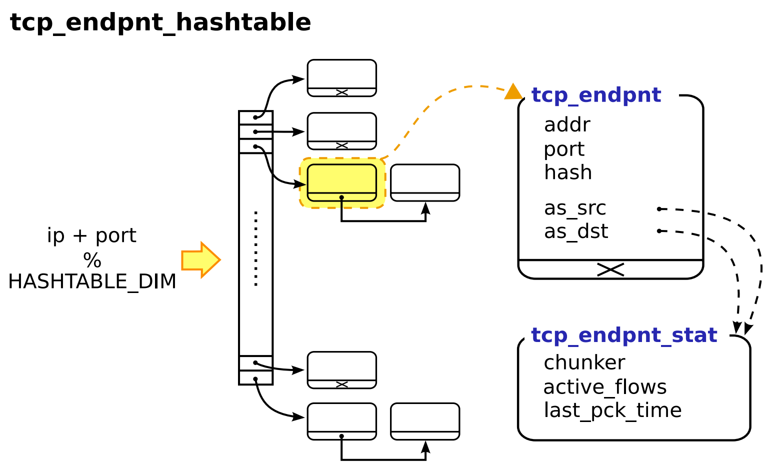


Figure 2.5. Tstat data structures managing endpoint statistics.

In order to manage also endpoint statistics some new data structures shown in figure 2.5 have been created. The core unit is `tcp_endpnt` being very similar to the previously seen `tcp_pair` structure which manages conversations and it is provided with two fields named `addr` and `port` permitting to specify the IP level network identifier of each endpoint. The link to the flow level structures is granted through `endpnt_a` and `endpnt_b` pointers defining the two hosts involved in the packet transmission. Statistics are stored through `as_src` and `as_dst` fields pointing to two different `tcp_endpont_stat` collecting respectively source side statistics and the destination side ones thanks to the `chunker` structure whose utilisation has already been illustrated. As a reminder it is stressed that, analogously to the flow level management, conversation direction specifies which structure has to be updated: if the current flow follows the direction *client*$\rightarrow$*server* then in `endpnt_a` must be selected the `as_src` structure and in `endpnt_b` the `as_dst` one, while if the direction is the inverse it should be done the countrary.
The `last_pck_time` field provides a temporal reference related to the last χ^2 update, while `active_flows` contains the number of active flows where the endpoint takes part as a source or a destination. These two previous fields have been thought to identify all flows in a long period idle state and to eliminate their related structures in order to free wasted space, presuming that for an unknown reason these associated flows have been closed or no more traced by Tstat. This fonctionality has not been

yet used, but it is available for future use. Any structure `tcp_endpnt` is collected and accessed through `tcp_endpnt_hashtable`, similarly explited as `ptp_hashtable`, with hash key equal to the sum of the endpoint id in module with the hashtable dimension.

2.3.2 Chunking structures

As said before the scope of the module is to divide a portion of some TCP packets payload in groups (*chunks*) of b bits and to collect frequencies of the assumed values in order to periodically generate χ^2 statistics vectors where each element is related to a specific group of bits. In order to do this some structures have been defined as shown in figure 2.6.

To simplify the chunker set up a configuration structure containing all working parameters has been created and it has been called `chunker_conf_fields` . It holds the following fields:

- `bits`, the number of bits each chunk is made up of;
- `bytes`, the bytes dimension of the payload portion which is analyzed;
- `direction`, chunk extraction direction (the default is reding the payload data string from the left to the right);
- `chunks`, max number of chunks of `bits` dimension achievable from the chunking of the portion with dimension `bytes`;
- `chunk_max_value`, the max number of distinct valus that can be represented by a group of `bits` binary symbols (always set up as 2^{bits});
- `enabled`, to state if this configuration is to be used or not.

The `enabled` field is present for the following reason: by considering the idea that any different setting is identified univocally by the chunk dimensioning and by limiting the max chunk dimension to 8 bits it is easy to figure out that there are only 8 possibile settings and it may be more confortable to have an interface for

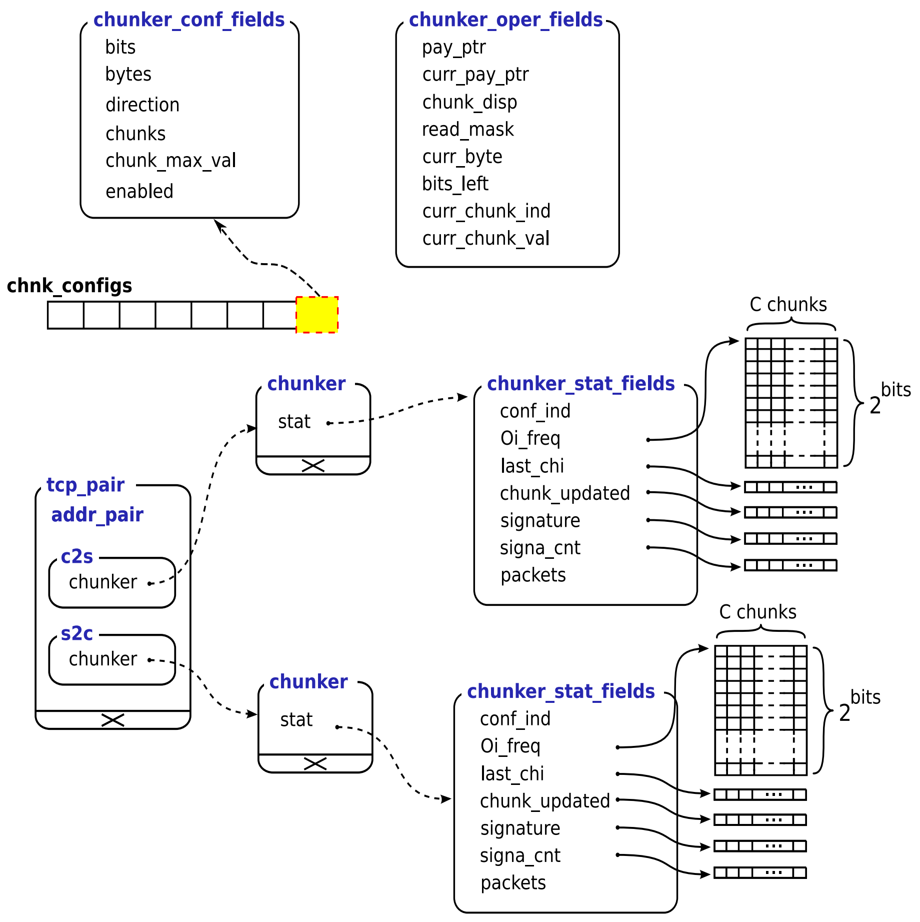


Figure 2.6. Data structures organization for the TCP-chunker module

easily select the good chunking configuration. So it has been decided to store in the `chnk_configs` vector all the possible `chunker_conf_fields` configurations. In this way set up is done just by taking into account that at position i in the vector corresponds the configuration with chunks dimension equal to $i + 1$ bytes. After this, `enabled` stands to figure out which one of the 8 possible configurations stored in the vector is used.
The `chunker_oper_fields` is used to ease the chunking extraction process and contains a list of variables useful to menage the payload content as it was a continuos stream of bits: the idea is to create a sliding mask over the stream flowing at the reading speed in order to have always the same number of available payload bits. The used variables are the following:

- `pay_ptr`, a pointer to the payload first byte;
- `curr_pay_ptr`, a pointer to the currently analyzed byte;
- `chunks_disp`, the max number of available chunks which can be yet extracted;
- `read_mask`, the bit mask allowing to extract a chunk from `curr_byte`;
- `curr_byte`, a copy of the byte pointed by `curr_pay_ptr` used as a window over the bit stream;
- `bits_left`, indicates how many bits in `curr_byte` have to be analyzed yet;
- `chunk_ind` and `chunk_val`, contain respectively the last chunk index and value.

The working principle is shown in figure 2.7.

Here the chunk dimension is supposed to be 3bits. At each extraction the value is passed to `chunk_val` and `chunk_ind` is incremented by one. while `curr_byte` content is shifted of 3 bits.

The `chunker_stat_fields` structure contains all statistics stored in matrices and vectors dinamically allocated accordingly to the setting parameters. It is composed of the following fields:

- `conf_ind`, position into `chnk_configs` vector where the `chunker_conf_fields` represents the current configuration. Used do acquire the dimensions of the statistics structures;
- `Oi_freq`, a matrix of `chunk_max_val` $\times$ `chunks` dimension where frequencies are stored in order to evaluate χ^2;
- `last_chi`, a vector of dimension `chunks` containing all the most recent χ^2;
- `chunk_samples`, a vector of dimension `chunks` containing the number of analyzed samples for any chunks;
- `chunk_updated`, a vector of boolean flags indicating if new samples have been extracted and statistics update status;

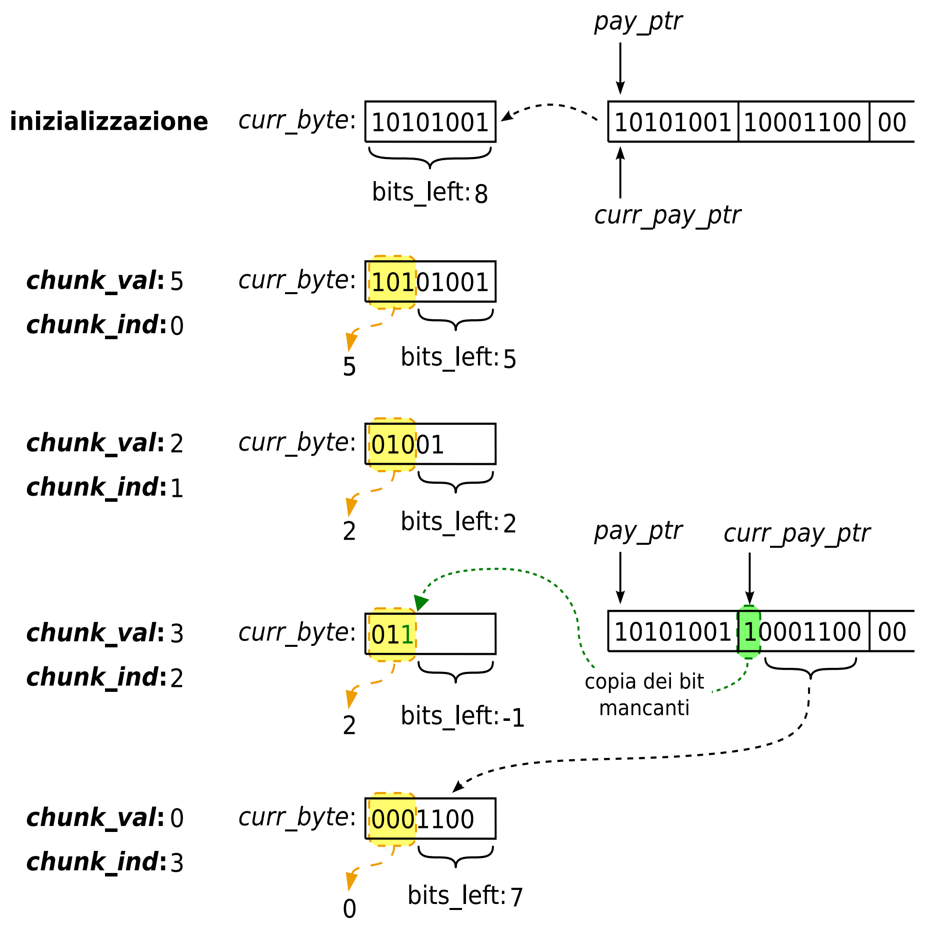


Figure 2.7. Chunking process example for the *left* → *right* direction by using 3bits chunks. `curr_byte` is the sliding window from where at each iteration the most significative 3bits are extracted. When there are not sufficient bits to create a new chunk, other bits are copied to the window by using the `curr_pay_ptr` pointer.

- `signature` e `signa_cnt`, two vectors containing respectively the last χ^2 vector produced and the related stability counter;
- `packets`, the number of packets already inspected;

Finally, the `chunker` structure is the one that organize all chunking statistics and in figure 2.6 it is depicted how it works as a container for the `chunker_stat_fields` structure and how both conversation directions (inserted in the `udp_pair` structure as shown before) have access to its own chunker statistics. In this way separated and independent statistics are obtained from each active flow and from each direction.

2.3.3 Chunking algorithm

The program's working scheme can be considered similarly to that of an iterator: packets are analyzed by the main function and little chunks are extracted one-one till the last available one. Then for each chunk a statistic χ^2 value is computed and, at regular intervals defined by the number of samples already collected, the statistics are dumped and stored in output files. Before the beginning of all operations `chnk_init()` is executed in order to set up and initialize some parameters, e.g. the previously shown vector `chnk_configs` containing all possibile chunking configurations.

In figure 2.8 it is shown how the chunker works and how it exploits its data structures in order to split the payload and to compute endpoint statistics. The elaboration starts with the `chnk_flow_stat()` function call executed within the plugin interface and it receives in input the next parameters:

- `pip`, a pointer to a structure containing the IP header fields used in order to compute the TCP payload as a subtraction between the IP payload and the TCP header;
- `pproto`, a pointer to a structure containing the TCP header fields;
- `tproto`, a value indicating the transport level protocol (not used in the module);
- `pdir`, a pointer to c2s or s2c structures according to the current flow direction. This ease the statistics computation and registering;
- `dir`, a value indicating the current flow direction (*client*$\rightarrow$*server* or *server*$\rightarrow$*client*);
- `plast`, a pointer to the end of the IP packet, used to compute the IP payload.

A first check is done and only the packets with a non null TCP payload are considered while the others cause the function to return. Then all initialization operations can start: the current chunker configuration is loaded from the `chnk_configs` vector and all current flow and endpoints identifiers are extracted. At this point a second check is performed on the number of already parsed packets belonging to the present flow so that only the first `MAX_TCP_PACKETS` are taken into account for the

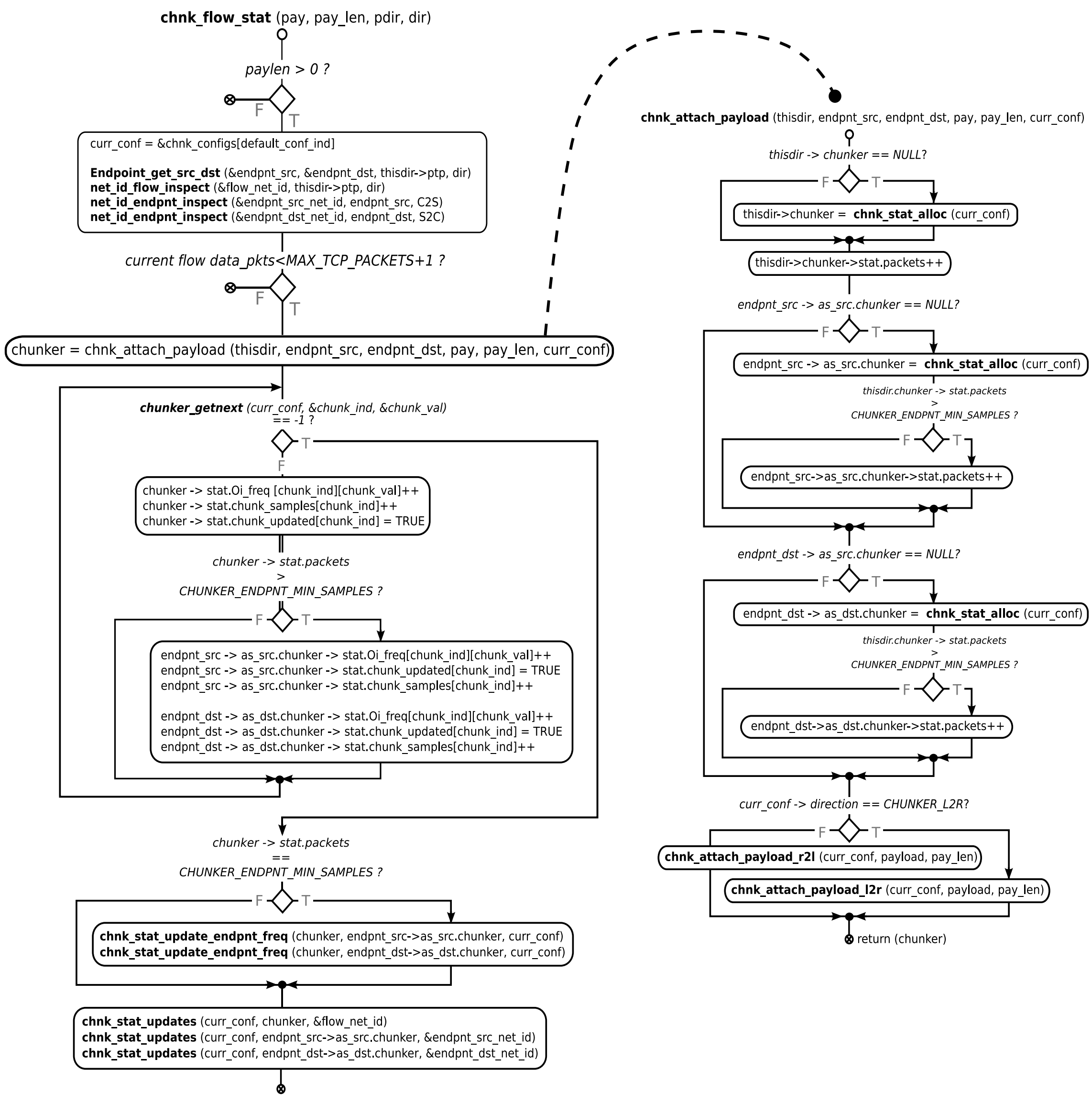

chnk_flow_stat (pay, pay_len, pdir, dir)
paylen > 0 ?
F
T
curr_conf = &chnk_configs[default_conf_ind]
Endpoint_get_src_dst (&endpnt_src, &endpnt_dst, thisdir->ptp, dir)
net_id_flow_inspect (&flow_net_id, thisdir->ptp, dir)
net_id_endpnt_inspect (&endpnt_src_net_id, endpnt_src, C2S)
net_id_endpnt_inspect (&endpnt_dst_net_id, endpnt_dst, S2C)
current flow data_pkts<MAX_TCP_PACKETS+1 ?
F
T
chunker = chnk_attach_payload (thisdir, endpnt_src, endpnt_dst, pay, pay_len, curr_conf)
chunker_getnext (curr_conf, &chunk_ind, &chunk_val) == -1 ?
T
F
chunker -> stat.Oi_freq [chunk_ind][chunk_val]++
chunker -> stat.chunk_samples[chunk_ind]++
chunker -> stat.chunk_updated[chunk_ind] = TRUE
chunker -> stat.packets > CHUNKER_ENDPNT_MIN_SAMPLES ?
F
T
endpnt_src -> as_src.chunker -> stat.Oi_freq[chunk_ind][chunk_val]++
endpnt_src -> as_src.chunker -> stat.chunk_updated[chunk_ind] = TRUE
endpnt_src -> as_src.chunker -> stat.chunk_samples[chunk_ind]++
endpnt_dst -> as_dst.chunker -> stat.Oi_freq[chunk_ind][chunk_val]++
endpnt_dst -> as_dst.chunker -> stat.chunk_updated[chunk_ind] = TRUE
endpnt_dst -> as_dst.chunker -> stat.chunk_samples[chunk_ind]++
chunker -> stat.packets == CHUNKER_ENDPNT_MIN_SAMPLES ?
F
T
chnk_stat_update_endpnt_freq (chunker, endpnt_src->as_src.chunker, curr_conf)
chnk_stat_update_endpnt_freq (chunker, endpnt_dst->as_dst.chunker, curr_conf)
chnk_stat_updates (curr_conf, chunker, &flow_net_id)
chnk_stat_updates (curr_conf, endpnt_src->as_src.chunker, &endpnt_src_net_id)
chnk_stat_updates (curr_conf, endpnt_dst->as_dst.chunker, &endpnt_dst_net_id)
chnk_attach_payload (thisdir, endpnt_src, endpnt_dst, pay, pay_len, curr_conf)
thisdir -> chunker == NULL?
F
T
thisdir->chunker = chnk_stat_alloc (curr_conf)
thisdir->chunker->stat.packets++
endpnt_src -> as_src.chunker == NULL?
F
T
endpnt_src -> as_src.chunker = chnk_stat_alloc (curr_conf)
thisdir.chunker -> stat.packets > CHUNKER_ENDPNT_MIN_SAMPLES ?
F
T
endpnt_src->as_src.chunker->stat.packets++
endpnt_dst -> as_src.chunker == NULL?
F
T
endpnt_dst -> as_dst.chunker = chnk_stat_alloc (curr_conf)
thisdir.chunker -> stat.packets > CHUNKER_ENDPNT_MIN_SAMPLES ?
F
T
endpnt_dst->as_dst.chunker->stat.packets++
curr_conf -> direction == CHUNKER_L2R?
F
T
chnk_attach_payload_r2l (curr_conf, payload, pay_len)
chnk_attach_payload_l2r (curr_conf, payload, pay_len)
return (chunker)

Figure 2.8. Chunker module flow diagram. `chnk_flow_stat()` receives a TCP packet: if it has a non null payload and if the number of already analyzed packets belonging to the same flow is less than N, all pre-chunking operations start and `chnk_attach_payload()` is called to set up operative variables. Then after each call of `chunker_getnext()` a new chunk is obtained and `chnk_stat_updates()` is used to generate χ^2 vectors with the proper periodicity.

statistics computation and the other ones cause the function to return. Then the function `chnk_attach_payload` is executed to allocate new `chunker_stat_fields` structures if the received packet represents the first one in a flow direction and to extract the set up parameters needed to split the payload and it inserts in the previously seen `chunker_oper_fields` structure all the operating parameters. So `chnk_getnext()`, working as described in the previous section, outputs the value and the index of a new chunk by copying them into the `chunker_oper_fields` proper fields or, if the available chunks are finished, it returns -1. Since this operation must be repeated a lot of time, this chunking function has been set as exit condition of an iterative cycle with the aim of extract all available chunks till the -1 value is given. At this point all chunks values are stored and frequencies updated in the flow chunker structures, waiting to pass the threshold allowing endpoint aggregation. When the number of analyzed packets belonging to the current flow is sufficient, frequencies are updated in the endpoint level chunker too. This threshold mechanism is needed because frequencies at endpoint level are based on those originated by distinct flows and then aggregated periodically: in this way it is possibile to obtain a noise level caused by short flows that in the worst case can represent a very significant portion of the network traffic and, after aggregation effect, they can affect deeply computations. It is compulsory to avoid this problem by using the `CHUNKER_ENDPNT_MIN_SAMPLES` threshold which delays all frequencies updates till the current flow has not yet reached the minimum packet number.

2.4 Data processing module

Tstat outputs two log files containing the χ^2 values written down so that on every row there is a specific value for a given chunk and a given endpoint following the format

```
<network id> <# of packets> <chunk index> <# of samples> <chi value>
```

This is due to the fact that statistics update is done periodically, monitoring the number of samples extracted for every chunk. Furthermore it is also taken into account the possibility that a given packet has not got a sufficient number of bytes so that it is not possible to extract all the desired chunks. This characteristic does not allow to have immediately a multicolumn tabular file showing any complete vectors associated to any endpoint.

The format of training and testing data file to feed the SVM module must be

```
<label> <index1>:<value1> <index2>:<value2> <index3>:<value3> ...
```

that in this case becomes

```
<proto label> 1:<chi chunk 1> 2:<chi chunk 2> 3:<chi chunk 3> ...
```

It is evident that it needs to pass through some steps to arrive at the desired format. In the first place all rows related to the same update step and to the same endpoint are written in the same line so that every whole vector is on a single row. After having removed useless rows not containing all chunks (these are computed over flows with payloads size not sufficiently large), endpoints are replaced with labels of the applicative protocol they are carrying. This has been done through the use of an oracle which associates any endpoint with its protocol. More detailled informations about its implementation are available in section 3.1.3.

2.5 LibSVM libraries

LibSVM [21] is a software created in order to ease and spread the use of SVM classifiers and it consists of a rich set of tools for both classification and regression. It provides an easy-to-use kernel library with the most used functions such as linear, polynomial, sigmoid and RBF gaussian. In order to limit computational times in multiclass problems it put to use the one-against-one method described in section 1.2.4 and it is able to output the probabilities related to belonging to each class. It's written in C language but interfaces are provided in Python, Java, R, Matlab, C#, etc.

A special tool is present in the LibSVM package constisting of a script able to menage all the tranining and testing phases with a minimum effort by the user and well introduced in [8]. Basically it requires training data to be expressed in the correct SVM format and it works with the following procedure:

1. it conducts scaling on the data;
2. it selects the kernel function;
3. it uses a grid-search to find the best parameters C and γ and a cross-validation training;
4. it performs the prediction over the testing set in input.

The correct data format for both training and testing sets is granted by the python module previously introduced in section 2.4, so the SVM receives already labelled vectors of χ^2.

1. Scaling

The first operation is the scaling one, acted to avoid that attributes in great numeric ranges dominate those in smaller ones and cause them the lose signifiance, and also to avoid numerical difficulties caused by too large numbers during calculation. The chosen range is [-1, +1] and it is obviously applied to training and testing sets both.

2. Kernel selection

The kernel function selection is strictly related to the type of dataset analyzed. In fact, for example, the RBF gaussian kernel can handle the case when the relation between class labels and attributes is nonlinear, a capability not owned by the linear or the polynomial kernels. In addition it has been proved in [17] that the linear kernel is just a special case of the gaussian kernel and in [18] that the sigmoid one can't behave better than the RBF. Another advantage to use this kernel is the limited number of parameters, which would be greater if considering the polynomial kernel, and this would cause difficulties in seeking the good parameters configuration.

3. Parameters set up

Since the chosen kernel is the RBF two only parameters have to be selected: C, the error penalty proper of the SVM with non-rigid solution seen in section 1.2.3 and γ, being the RBF parameter. The best pair (C,γ) is used, the more accurated will be the model generation. Since it is not known a priori which pair is optimal, a *grid-search* is carried out trying exponentially growing sequences of C and γ. Then, for each pair under test, a *v-fold cross-validation* is accomplished in order to prevent the overfitting and so the training set is firstly divided into v subsets (in the next chapter tests $v = 5$ has been selected) of equal size and than, in turn, each subset is tested using the classifier trained with the remaining $v - 1$ subsets. In this way every object of the training set is predicted once and used for training $v - 1$ times; the accuracy is computed as the percentage of correctly classified data. Finally the

pair (C,γ) with the best cross-validation accuracy is selected.
To this end LibSVM provides an useful tool using a gnuplot extension: after completion of the grid-search a graph is available showing the trend of the accuracy respect to the pair chosen: in this way it can be done a finer search with a more limited range.
In the classifier here implemented this kind of strategy has been used since is not too much computationally expensive since the grid-search can be parallelized because each pair is independent and since this exaustive method has acceptable times if used with two parameters.

4. Prediction

Finally, after the learning phase in which the model is completed, prediction is done on the testing set. This file is composed, exactly as the training set, of a large number of χ^2 vectors provided of their own label: this is done in view of the validation phase since the SVM can compute the accuracy of the prediction. This can be an interesting element, though coarse enough, to fastly analyze prediction results before carring on more accurate studies over them.

Chapter 3

Empirical results

3.1 System setup

After having developped the whole model and tested separately every modules composing it, several traffic analyses have been performed in order to evaluate prediction performances in some different situations.
First of all, every chunking parameters have been correctly configured by setting up the chunk dimension to 4bits and the analyzable payload portion to 12bytes. In this way 24 chunks are obtainable from each packet payload big enough. These statistics are automatically written into two log files representing data aggregated by source and by destination, therefore providing separated informations about the two flow directions. Another important parameter, as discussed in section 2.3.3 is represented by the number of packets to analyze having a non-nul payload for each flow. This has been modified during the tests to find out how it affects performances changing it from 5 to 3 packets per flow.

In order to obtain accurate χ^2 computations it has been decided to set the statistics update step to 80 samples for each chunk. As already justified in 1.1, this periodicity is given by the probability theory, in fact to obtain a reliable χ^2 value it must be observed a number of samples equal to 6-7 times the cardinality of the possible values which the variable can assume. That means that since the size of each chunk is set to 4bits, 16 different values are obtainable and this implies the need of about one hundred samples. Anyway, deterministic and mixed cases are also to be considered, and this decreases for sure the number of samples to wait before computing statistics, so for the current case study the step value has been set to 80, and this does not affect the system reliability at all.

This decision of using such a threshold has strongly influenced the statistics elaboration mechanism, in fact it has been observed a loss of long and voluminous enough flows, that's to say that the most of traced communications is composed of less than 80 packets: this implies that there are not enough samples for each chunk in order to produce per flow signatures. To solve the problem it has been decided to analyze the traffic at an endpoint level, so chosing of aggregating packets (and so statistics), for each different host involved in a flow communication.

Another reason to use such a 80-packets' step is to obtain an *oversampling* benefit: in order to achieve a great number of χ^2 vectors after every 80 packets all the structures containing statistics are re-initialized and all old values are erased. So computations start again and new vectors are obtained. In this way the dataset contains a signature for each endpoint and for each 80 packets' sequence belonging to the same flow, and since the χ^2 has an incremental nature and rises its value after any update, the problem of coping with big values is avoided because it is periodically reset to 0.

In figure 3.1 it is shown the data process developped in order to test classification performances. Initially the trace is filtered according to some policies better described in the next pages and the oracle, representing the "truth teller" about the actual classification, is instructed. Then splitted applicative traces are separately processed by the Tstat TCP-chunker which creates signatures so that they are sampled and randomly splitted into a testing set and a training set. After the creation of the model file starting from the training set, the classification is done on the testing set and a file containing all previsions is output, ready to be compared with the oracle which is now able to verify the correctness of predictions just performed.

3.1.1 Dataset

The traffic trace object of study has been captured by a probe placed at the edge router of the Politecnico di Torino network so that all incoming and outgoing traffic on the access link was considered for analysis. In figure 3.2 is presented a scheme depicting how traffic measurements have been performed. In particular, a DAG

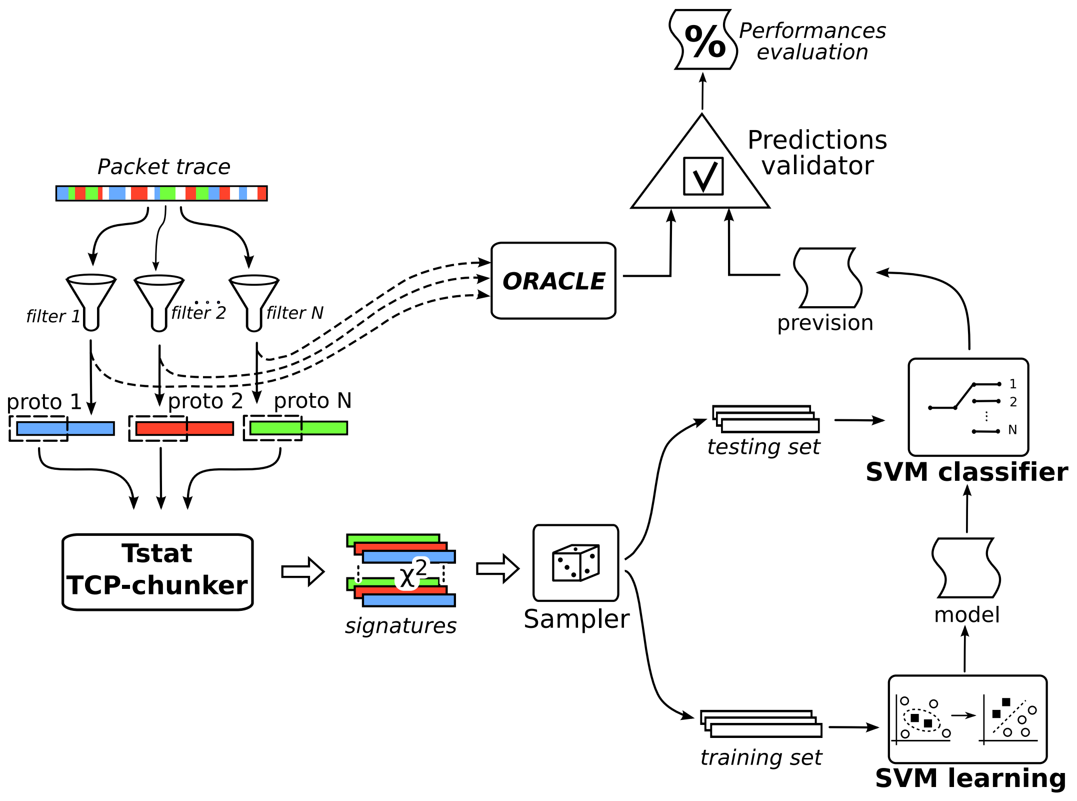


Figure 3.1. Scheme resuming the trace elaboration process: after filtering the trace and instructing the oracle, the protocol traces are processed by the Tstat TCP-chunker in order to obtain χ^2 signatures representing the whole dataset. Then after a careful random sampling the dataset is splitted into a training set and testing set: the first one feeds the SVM and allows it to produce the model used to classify testing objects. Finally the assigned labels veridicity is checked by the oracle and that permits to give a performances evaluation.

card has been installed on a PC connected to the edge router of the POLITO network so that data are collected between the network border and the access router of GARR/B-TEN, the Italian and European Research network. The LAN contains about 7000 hosts, part of them are servers regularly accessed from the outside but the most are clients whose users can be administrative, faculty members or students, and most of the traffic is due to TCP data flows carrying web, email and bulk traffic. The backbone of the campus LAN is based on a switched Fast Ethernet infrastucture providing one only interface for access to the GARR/B-TEN network and, through it, to the public Internet.

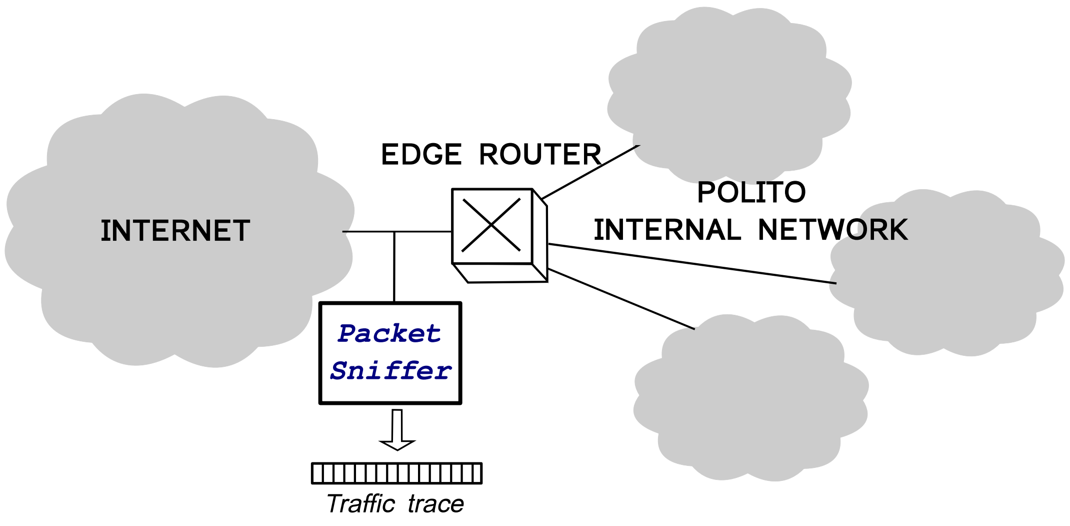


Figure 3.2. Scheme of the network where traffic analysis have been performed

The analyzed trace is a 110 GB file "cut" by a greater dump produced after a four months long sniffing activity and it contains a huge variety of protocols and traffic types. Among all these, it has been decided to take into account only some more relevant typologies so that classification has been done trying to identify 10 categories of TCP traffic:

- FTP: a *File Transfer Protocol* for exchanging and manipulating files placed on a remote server;
- SSH: *Secure SHell*, a network protocol that allows data to be exchanged using a secure channel between two networked devices. It uses a public-key cryptography to authenticate the remote computer and allows the remote computer to authenticate the user;
- SMTP: *Simple Mail Transfer Protocol*, it is nowadays a standard for e-mail transmissions across the Internet. As it is text-based, the message is transferred using a procedure of queries and responses between the client and server. Electronic mail server software uses SMTP to send and receive mail messages, user-level client mail applications typically only use SMTP for sending messages to a mail server for relaying;

- HTTP: *Hypertext Transfer Protocol*, the well known communication protocol for the transfer of information on the Internet. It is a request/response standard between a client and a server, the client being the end-user and the server being the web site;

- POP: *Post Office Protocol*, used by clients to retrieve e-mails from a remote server;

- IMAP: *Internet Message Access Protocol*, is another prevalent Internet standard protocol for e-mail retrieval;

- SKYPE: a well-known proprietary communication protocol allowing users to make telephone calls over the Internet. A lot of research studies have been carried out in order to be able to identify it (interesting results are contained in [19]).

Furthermore it has been tried to identify some of these protocols while protected with a secure connection, so it is possibile to add HTTPS, POP3S and IMAPS to the previous list.

To improve the system classification sensibility it has been decided to consider also a generic *background class* containing all the residual traffic, that's to say all other application protocols that are not considered in the already listed classes. This idea is justified by the fact that the wholeness of filtered traces represents only a fraction (about 1/8) of the original trace. This means that the most of the traffic doesn't take part in one of the service classes. It's then clear that a complementary class is demanded in order to collect all this remaining non-preclassified traffic which, if not, would have soiled the system performances trying to be associated to a class which would be for sure not the correct one.
This background class has been derived by filtering again the trace in a complementary way respect to the previous times: the filter represents the negation of the union of all filters used to define service classes, but with some simplifications. The filtering process carried out is described in details in section 3.1.3.

3.1.2 Trainset populating policies

In order to have the most correct and effective prediction capability, a good SVM training process is essential so that a well fitting classification model is elaborated. So the training set must be accurately arranged.
The first parameter to consider is the dimension of the training set: it must be big enough to grant a good classes characterization but if it is too big it can cause overfitting problems, meaning that the classifier is so much specialized on the training that when analyzing samples from the testing set it is not able to identify them with effectiveness. That is the reason why a big and detailled training set doesn't lead automatically to improve performances, but on the contrary it can easily be cause of unaccuracy.
Since the main aim is to obtain an homogeneous and uniform training set, among all the samples contained in the dataset a *sampling* process is executed and the most representative data objects are selected. This sampling operation is performed with a uniform distribution probability developped on two levels: firstly among all endpoints one is picked with a uniform probability and then, always uniformally, a statistics vector associated to the extracted endpoint is selected. In this way the problem of an unbalanced training in advantage of a limited number of endpoints with a large number of signatures associated is avoided and it can be said that the system has a balanced and homogeneous training phase. It has been also tried to force that an endpoint could appear in the training set only once, but this has not been possible since several classes didn't have a sufficient number of distinct host ids.
Several tests have been devoted to the pursuit of a good training set dimensioning, always changing the balances among different classes in order to improve the whole accuracy. In fact it is clear that if the classifier is fed with a lot of samples belonging to a class and only a few related to another one, performances would get worse since the second kind of objects would not be identified and this would result in an augmentation of False Positives related to another class too. Therefore, due to the great number of different classes specified in the tests performed, it has not been easy to chose a good populating policy since a lot of changes in single classes populations have been demanded.
The first observation which has been made is that a *pure proportional policy* is not at

all effective, meaning that if a class' training set is raised with a number of samples always proportional to the class dataset cardinality, this brings to a low accuracy. Formally this condition can be written as

$$\frac{\#\ \textit{training set proto 1}}{\#\ \textit{dataset proto 1}} = \frac{\#\ \textit{training set proto 2}}{\#\ \textit{dataset proto 2}} = ... = \frac{\#\ \textit{training set proto N}}{\#\ \textit{dataset proto N}}$$

This proportionality must be tared according to the less numerous class in order to assign it at least one element in the final training set and all the other cardinalities must be computed always maintening constant this ratio. The problem with this policy is that the less populated classes are not sufficiently characterized and so their samples cannot be correctly classified. On the other hand the most populated classes achieve high False Positives rates since they uncorrectly absorb most of sample belonging to the not so characterized classes. This is true in particular for the *background* class whose irregularity and etherogeneity causes the amalgamation of unrecognized traffic.

After sperimenting the inefficiency of the pure proportional policy another approach has been put under test and it can be familiarly called the *balanced* one. It consists of populating traning classes in a progressive fashion so that classes related to datasets with the same order of magnitude are raised with quite the same number of test samples and all other classes cardinality changes progressively. In this way also the less numerous protocols can be sufficiently represented in the training set and the disproportion with the larger datasets is reduced. The disadvantage of using this mechanism is that the search of a good samples distribution requires a lot of tests (and so a lot of time) and so the optimum solution is not obtainable, at least in a reasonable time.

3.1.3 Trace filtering and Oracle definition

As explained at the beginning of the current chapter, considering only TCP packets, the initial trace is firstly filtered trying to split it according to the applicative protocol: this is done in order to be able to assign a correct protocol label to all samples in view of the validation phase.
Excluding for the time being the skype protocol, two kinds of filtering have been performed:

- port-based filtering;

- endpoint-based filtering by means of a *DPI* analysis.

Port-based service filtering

This method goes on the basis of the old classification paradigm stating that each protocol (at least the most known) has an own dedicated port which is always used. So an association between service and port can be done:

- FTP: 21
- SSH: 22
- SMTP: 25
- HTTP: 80
- POP3: 110
- IMAP2: 143
- HTTPS: 443
- IMAPS: 993
- POP3S: 995

It's clear that this can't represent a satisfying classification policy since there is a variety of new Internet applications that either do not use well-known port numbers or use other protocols, such as HTTP, as wrappers in order to go through firewalls without being blocked. In addition, emerging services avoid the use of well-known ports probably to avoid detection, e.g. some peer-to-peer applications.
These are the reasons because this kind of filtering has just been used for a preparatory study supporting the classification issues described in this work, therefore a port-based approach has been used only in the first tests.

Endpoint-based service filtering

This is a more complex method which should provide a reliable service pre-classification. It is based on the use of a *DPI - Deep Packet Inspection* tool that examines the data and header part of any packet. In particular `tshark`, the command version analogous to Wireshark (ex. Ethereal), is a network protocol analyzer able to perform a deep inspection into each single packet and to filter a chosen service. The inspection consists of a particular pattern matching in the application layer data: if a string identifying the protocol is found, the packet is filtered.

In this way it is possible to obtain all the packets containing the service data and including the specific pattern, but this is yet not sufficient in order to extract from the initial trace all the service flow packets. So the idea is to store the endpoints involved in the packet transmission and re-filter the trace by endpoints: it is evidently a high computational cost operation, but it has been tested that if splitting the original filter in several littler ones and launching them in concurrence on the trace than the process is smoother and less resources consuming. It must be noted anyway that these operations have been performed by means of a eight-core computer with an Intel Xeon 2.33 GHz processor and 4096GB RAM so that they didn't take an excessive amount of time.
Another consideration to discuss is that it has seemed to be useless and probably a failure in processing to make an endpoint filtering to the whole 110GB trace, without taking into account that this filtering would have been repeated for all the defined protocols.

For all these motivations it has been decided to apply the endpoint-based filtering to a port-based pre-filtered trace, being on the opinion that in this way only pure protocol traffic is obtained and all those flows carring the given protocol tunnelled on another port are negiglible and can be not considered. In figure 3.3 the whole filtering process in order to extract a protocol traffic is shown, the operation is obviously to repeat for all protocols.

Skype traffic filtering

Since Skype exploits a proprietary and encrypted protocol it is very difficult to identify packets belonging to that. Hosts chose actively which port to use for the communication and this is not predictable a priori, and on the other hand since the traffic is strongly encrypted there is no way to perform a Deep Packet Inspection in

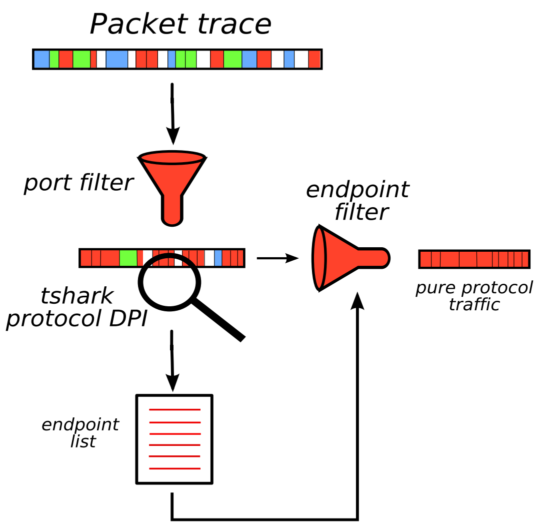


Figure 3.3. Filtering process using both port-based and endpoint-based approach trough DPI. The trace is initially filtered by the service standard port and later re-filtered by tshark using DPI in order to look for only packets carring protocol data and all the involved endpoints are stored in a list. Finally the trace containing only traffic on the standard port is filtered again according to an endpoint-based approach using as filter the list of endpoint derived in the previous step.

order to search a certain pattern in some known application header field. Anyway the study of this protocol has stimulated the interests of the scientific community and several methods have been developed in order to be able to trace skype transmissions. The filtering mechanism used here to isolate the protocol has been derived by Politecnico di Torino in cooperation with TELECOM ParisTech and it bases on a randomness study performed on sequences of bits in order to detect the packet's framing structure (see [19] for further informations).

Thanks to this robust classification method the list of endpoints involved in skype communications has been available and so an endpoint-based filtering of the whole trace has been performed and TCP skype traffic effectively isolated. Basically it is made up of signalling and data messages while all voice communications are carried through UDP and therefore not concerned in this classification analysis.

Background traffic extraction

After having produced all the protocol traces, there is yet one class remaining to be defined and this is the background one. As said before it is composed of all the traffic not belonging to the already defined classes and for this it can be thought as a residual trace.
As foremost thought the class should be populated by the exact complementary subset of all the others' union in order to obtain an exact partition of the total set. Unfortunately this approach, though hypothetically correct and feasable, leads to huge computational costs and an unexpected series of practical problems due to the trace size, the filter (ultra)complexity and the memory usage limitations imposed by filtering tools like `tcpdump` and `tstat`.

For this reasons the first idea has been abandoned to be supplanted by a simpler approach based on the hypothesis that just a port-based negative filtering should have been sufficient to remove all the known and already classified traffic from the background subset. In this way the original trace has been purged by all the standard ports and by means of a final endpoint-based negative filtering in order to remove the skype traffic too.
It's clear that in this way a certain amount of non applicative traffic is cut out since as discussed before the port-based filtering is not completely effective in isolating a given service protocol, but this can be seen as a conservative simplification which doesn't affect the class partition consistency.

Oracle definition

After filtering the trace and producing N different service traces ready to be analyzed by Tstat, the Oracle has to be created and instructed. It consists of a list of endpoints to which a protocol label is associated: in this way it is evident how every endpoints are involved in an applicative communication and this permits an easy validation. In fact the TCP-chunker produces signatures related to each single endpoint and, before submitting them to the SVM they should be correctly labelled: the Oracle is appointed to do this, by replacing the endpoint id with the proper protocol label (which is merely the port number) in order to make a fast confrontation between the training test file labels and the prediction file ones.

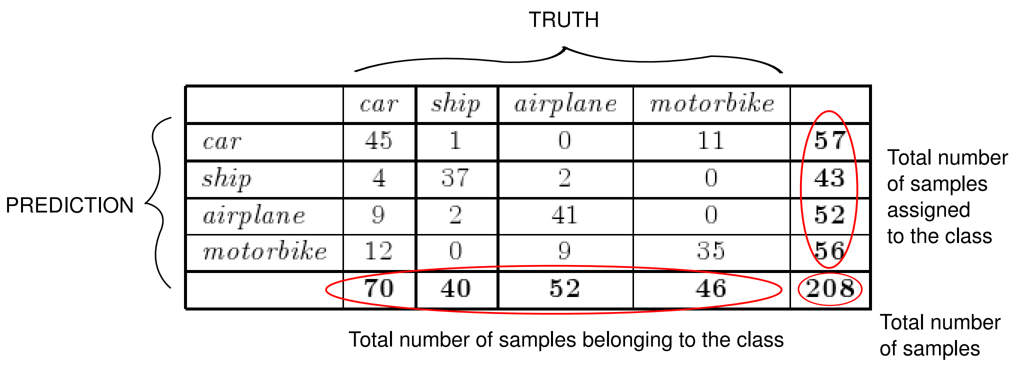
3.2 Classification performance

In this chapter, after a brief explication about the performances measurement indices used during the work, all results are shown. Several tests have been carried out starting from a preparatory study characterized by a port-based oracle with 24 chunks long signatures covering a 12 bytes portion of each payload, followed by an analysis on the number of packets to consider for each flow, to go on with some trainset dimensioning tests. Then, trying to develop the idea that also TCP flags and options could help in classifying traffic, 5 more chunks containing only signalling bits have been added to signatures and new tests have been performed.
Finally every concepts and optimisations tricks found out during testing sessions have all been used to set up a last testing configuration providing very satisfactory results.

3.2.1 Information retrieval issues

Before starting analyzing the achieved testing results it is compulsory to focus on the established measurement methods in order to deeply understand the meaning of the next chapters data representations.

Classifier performances are measured through the use of the so called *confusion matrix*, a method reporting at a glance the number of True Positives and False Positives generated by previsions. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class, allowing to see if the system is confusing some classes and in what ratio.
As an example, a simulated classification result is displayed in figure 3.4 showing how a generic classifier recognize four different means of transport. Numbers in bold on the right of the table, computed by summing up the quantities on the same row, indicate how many elements have been assigned to the class related to that row, while those at the bottom point out the number of samples actually belonging to the class of the column. On the diagonal there is the number of True Positives, meaning the samples well classified, while all the other values represent False Positives, being all the erroneously classified samples. For example, among the 208 samples used during the test, 45 have been exactly classified as cars, 37 as ships, 41 as airplanes and 35 as motorbikes while the hardest misclassifications are represented by the 11


TRUTH
PREDICTION
Total number of samples assigned to the class
Total number of samples belonging to the class
Total number of samples

	car	*ship*	*airplane*	*motorbike*	
car	45	1	0	11	**57**
ship	4	37	2	0	**43**
airplane	9	2	41	0	**52**
motorbike	12	0	9	35	**56**
	70	**40**	**52**	**46**	**208**

Figure 3.4. Performance presentation example: means of transporting identifier. The total numbers of assigned samples and of the actual samples for each class are also pointed out in the table.

actual motorbikes mistaken for cars, 12 cars erroneously identified as motorbikes, 9 airplanes mistaken for motorbikes. After these considerations an interesting element to be found out is the *accuracy* parameter computed as

$$accuracy = \frac{number\ of\ True\ Positives}{number\ of\ True\ Positives + number\ of\ False\ Positives}$$

meaning the ratio between the number of correctly classified samples (the sum of values on the main diagonal) and the total number of samples used in the test. So a 100% accuracy indicates a perfect classifier able to distinguish every object in the test set. Anyway, in the means of transport example the accuracy is 76% and this means that one out of four objects are not correctly identified, so the classifier is not sufficiently reliable. It's clear that this quantitative analysis is not so meaningful since it is not carried on with respect to the total number of samples and a more interesting element could be to know the classification error ratio for a class according to its total number of elements and how the system confuses objects. So in table 3.1 percentage results are shown emphasizing the system classification capabilities by displaying for every cell in the previous table the percentage of the related number with respect to the total number of samples for that class (computed summing up all the current column values). These data are then useful to understand in what percentages samples actually belonging to a class are assigned to a label after prediction. After this further data mining it is now evident what are the stenghts and the weakness in the system: ships are quite well classified since the False Positives ratio is low while for all other classes, especially the car one accusing

	car	*ship*	*airplane*	*motorbike*
car	64,29	2,5	0	23,91
ship	5,71	92,5	3,85	0
airplane	12,86	5	78,85	0
motorbike	17,14	0	17,31	76,09
	100	**100**	**100**	**100**

Table 3.1. Classification percentages for the means of transport example

a great problem in its identification, the classifier points out inefficiency. In fact the system erroneously classifies as cars the 24% of motorbikes under test and can't correctly identify the 36% of actual cars.

3.2.2 Preparatory tests

In its first version the classifier couldn't achieve optimal results since it has just been easily set up in order to test if all mechanisms and modules worked correctly and if the χ^2 metrics gave a good feedback.

First and foremost all chunking parameters have been inserted in the Tstat module being in favour of starting with an analysis confined to 5 packets per flow having a non-nul payload and to extract 4bits chunks from the first 12bytes of the payload, obtaining a total of 24 chunks. The statistics update and reset steps have been both set to 80 samples per chunk according to what discussed previously.
With regard to the dataset it has been decided to populate it using signatures produced after a port-based filtering: 10 approximate service traces have been derived filtering by standard port as described in section 3.1.3, except for the skype traffic which has been extracted with the method described before. The *other* class contains the background traffic consisting of all the packets not addressed to and not originated by one of the standard ports related to a class. During all these preparatory tests it has been used always the same dataset, consisting of 779.018 samples for the destination aggregation case and of 535.663 samples for the source aggregation one, respectively related to 62.321.440 and 42.853.040 packets analyzed.

Having set this configuration, some preliminary tests have been performed and a first training size calibration has been carried out, demonstrating how a proportionally populated set does not achieve satisfying results. Table 3.2 shows this samples distribution applied in order to build the training set and in tables 3.3 are presented the related results for a 10-protocols classification, dividing the source and the destination aggregations cases. The worst False Positive rates are highlighted in red in order to emphasize the system bad functionings and to ease comments.

Table 3.2. Pure proportional samples distribution used in the SVM training operation

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other	
DST	903	227	5	157	1	1	14	2	28	20	3660	5018
SRC	1071	385	3	325	1	1	26	3	34	33	3111	4993

Table 3.3. Testing performances for a proportionally distributed training after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	71,26	0	0	0	0	0	14,25	0,66	0	0,06	4,87
https	0,02	76,37	0	0	0	6	0,55	0	0	0,32	1,2
ftp	0	0	38,5	0	0	0	0	0	0	0	0
smtp	0	0	0	99,85	0	0	0	0	0	0	0,04
imap2	0	0	0	0	0	0	0	0	0	0	0
imaps	0	0	0	0	0	0	0	0	0	0	0
skype	0,36	1,17	0	0	1,54	0	37,73	0	3,1	0	1,12
ssh	0	0	0	0	0	0	0	65,12	0	0	0,04
pop3	0	0	40,71	0	0	0	0,07	0	71,14	0	0,13
pop3s	0	0,21	0	0	0	48	0,02	0	0	90,94	0,06
other	28,36	22,26	20,8	0,14	98,46	46	47,37	34,22	25,76	8,69	92,54
	100	100	100	100	100	100	100	100	100	100	100

From the performances tables it is possible to infer that aggregating statistics by destination endpoints the classification is accomplished in a more accurated fashion.

Table 3.3. Testing performances for a proportionally distributed training after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	48,89	0	0	0	0	0	2,74	0	0	0	3,4
https	0,45	70,78	0	0	0	2	0,05	8,17	0	1,3	0,54
ftp	0	0	5,91	0	0	0	0	0	0	0	0
smtp	0	0	0	97,47	0	0	0	0	0	0	0,19
imap2	0	0	0	0	0	0	0	0	0	0	0
imaps	0	0	0	0	0	0	0	0	0	0	0
skype	0,03	0,04	0	0	6,15	0	12,58	0	0	0	0,77
ssh	0	0	0	0	0	0	0	20,92	0	0	0
pop3	0	0	0	0	0	0	0	0	10,64	0	0,02
pop3s	0	0,61	0	0	0	77	0	0	0	82,58	0,01
other	50,63	28,56	94,09	2,53	93,85	21	84,64	70,92	89,36	16,12	95,08
	100	100	100	100	100	100	100	100	100	100	100

In fact in the former case the obtained accuracy is 79% while in the latter case study it is about 71%. In both situations it is evident how the *other* class absorb lots of samples so generating a huge amount of False Positives, and this is due to the presence in every pre-classified service traces of a non-negligible amount of traffic which actually does not belong to the class at issue and that is also present in great quantitatives in the *other* class. So the SVM perceives the same kind of traffic and this brings to high False Positives rates.
Another aspect to stress is how any of the 65 *imap2* samples or the 100 *imaps* objects are not identified, regardless to the direction analyzed. This is due, how already said, to a lack of characterization for these protocols since only 1 class-related sample has been inserted into the training set: in fact all *imap2* traffic is identified as background, while *imaps* is mostly mistaken as *pop3s* probably because they use the same SSL encryption.
The source-aggregation case gives the worst results: the most part of service traffic is absorbed in the *other* class. Since the accuracy rate is computed on the number of samples correctly classified, this case obtains a quite high performance only because

439496 out of 771158 samples (so a half) belongs to the *other* class.

After the first trial with a proportional training set, many different changes have been performed in the samples distribution in order to improve accuracy and minimize False Positives. The best training set found is that described in table 3.4 where datasets cardinality is not taken into account in a proportional fashion, but in a progressive way, in order to provide to any class a sufficient amount of training samples such that the SVM is capable of effectively extracting any class traffic characteristics.

Table 3.4. Balanced samples distribution used in the SVM training operation

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other	
DST	1260	470	130	157	35	50	1400	120	200	220	2562	6604
SRC	1575	500	310	389	40	55	1100	135	200	260	2562	7126

Table 3.5 provides classification results for the balanced training set. At first sight it is evident how the system discrimination capability is improved: protocols samples are not absorbed by the background class. Anyway, after a more careful analysis, the percentage of True Positive belonging to the *other* class is steeply decreased, causing a non improvement into both accuracy rates which remain at 79% for the destination aggregation case and even get worse till 69% in the source endpoint study.

Finally some considerations can be taken into account: the point at issue stands in the lack of consistency of the pre-classification and so to an unreliability of the oracle, since it is clear that port-based filtering is not able to split service traffic effectively. Even if a good training set dimensioning and balancing has helped the system to better recognize different protocols, performances remain still not satisfactory and this is proved by the low accuracy rates. Moreover, there is some difficulty to identify *skype* and *http*, and both classification and performances evaluation can't output safe results since their operativity strictly depends on the oracle precision. This matters can be solved only performing a more precise pre-classification in order to better instruct the oracle.

Table 3.5. Testing performances for a balanced training after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	76,19	0	0	0	0	0	18,92	0	0	0	7,88
https	0,01	89,84	0	0	0	0	1,1	0	0	0	2,38
ftp	0	0	95,96	0	0	0	0	0	0,03	0	0
smtp	0	0	0	99,52	0	0	0	0	0	0	0,04
imap2	0	0	0	0	80,65	0	0	0	0	0	0
imaps	0	0,02	0	0	0	90,2	0,03	0	0	0,24	0,01
skype	0,47	0,57	0	0	3,23	0	36,65	0	0,42	0	1,99
ssh	0,03	0	0	0	0	0	0,01	100	0	0	0,1
pop3	0	0	1,01	0	3,23	0	0,04	0	89,29	0	0,08
pop3s	0	0,22	0	0	0	9,8	0,01	0	0	99,3	0,05
other	23,3	9,34	3,03	0,48	12,9	0	43,24	0	10,27	0,46	87,48
	100	100	100	100	100	100	100	100	100	100	100

Table 3.5. Testing performances for a balanced training after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	71,47	0	0	0	0	0	15,22	0	0	0	12,51
https	0,43	82,89	0	0	0	0	0,33	0	0	0,04	1,85
ftp	0,3	6,11	90,86	0	0	0	3,52	0	0	0	2,66
smtp	0	0	0	98,95	0	0	0	0	0	0	0,04
imap2	0	0	0	0	53,85	0	0	0	0	0	0,1
imaps	0	0	0	0	0	91,3	0	0	0	0	0
skype	0,07	0,18	0	0	0	0	15,73	0	0	0	2,48
ssh	0,02	0	0	0	0	0	0,05	100	0	0	0,04
pop3	0	0	0	0	0	0	0	0	97,07	0	0,02
pop3s	0	0,52	0	0	0	6,52	0	0	0	99,86	0,01
other	27,71	10,3	9,14	1,05	46,15	2,17	65,15	0	2,93	0,11	80,3
	100	100	100	100	100	100	100	100	100	100	100

3.2.3 Adjusting the number of packets to analyze

After having done the first dimensioning trials, the dataset has been worked out again since it has been decided to try some changes in the number of per flow packets analyzed and chunked by Tstat. The new dataset has 480685 objects at the destination side and 658351 at the source one. The training set has been created again in a balanced fashion as done in the previous test so it is the same as that depicted in table 3.4.

Table 3.6. Testing performances of a *4 packets* analysis after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	80,32	0	0	0,04	0	0	21,33	0	0	0	7,74
https	0,02	85,39	0	0	0	0	0,37	0	0	3,12	2,14
ftp	0	0	100	0	0	0	0	0	0	0	0
smtp	0	0	0	98,84	0	0	0	0	0	0	0
imap2	0	0	0	0	95	0	0	0	0	0	0
imaps	0	0,35	0	0	0	93,33	0,02	0	0	0,19	0
skype	0,54	0,48	0	0	0	0	36,71	0	0	0	2,42
ssh	0,07	0	0	0	0	0	0,01	98,28	0	0	0,21
pop3	0,02	0	0	0	0	0	0,04	0	99,65	0	0,06
pop3s	0	0,12	0	0	0	6,67	0,02	0	0	96,24	0
other	19,02	13,67	0	1,11	5	0	41,5	1,72	0,35	0,46	87,42
	100	100	100	100	100	100	100	100	100	100	100

Results in table 3.4 show that setting the number of packets equal to 4 performances don't change too much from the $N = 5$ case, but it is really interesting to see how some protocols such as *http* and *pop3* clearly increase the amount of True Positives in both directions.

So it is possible to gather that some protocols are better identified by considering a certain number of flow packets because in such cases the very first packets give much more discriminating information. For example *http* traffic can be strongly identified by the triple way handshake, so a first three packets analysis should give even better results. And in fact after having set the number of packets equal to 3, a

Table 3.6. Testing performances of a *4 packets* analysis after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	78,21	0	0	0	0	0	12,54	0	0	0	11,2
https	0,44	86,95	0	0	0	0	0,43	0	0	0,78	1,67
ftp	0,35	7,06	98,73	0,01	0	0	4,16	0	0	0	3,13
smtp	0	0	0	99,66	0	0	0	0	0	0	0,02
imap2	0	0	0	0	30,77	0	0	0	0	0	0,3
imaps	0	0	0	0	0	92	0	0	0	0	0
skype	0,04	0,2	0	0	0	0	15	0	0,03	0	2,5
ssh	0,03	0	0	0	0	0	0,16	99,05	0	0	0,23
pop3	0	0	0	0	0	0	0,02	0	98,67	0	0,02
pop3s	0	0,4	0	0	0	8	0	0	0	99,22	0
other	20,94	5,39	1,27	0,34	69,23	0	67,69	0,95	1,3	0	80,91
	100	100	100	100	100	100	100	100	100	100	100

further improvement has been observed and the *http* True Positive percentage has raised till 82%. Entire results for the 3 packets trial has not been published since the less populated classes haven't been able to acquire a sufficient number of samples to make the tests. Anyway, except for *http*, the general trand is that performances are not improved, but at most they remain at the same level (*pop3*, *pop3s*, *https*, *smtp*). True Positives percentages collapse in *imap2* and *ssh*.
Another interesting behaviour is shown by *ftp* which improves its performance only in the SRC→DST (from source to destination) direction, while in the other one it collapses and this can be caused by the ease in recognizing the host protocol commands rather than server data trasmission.
A conclusive consideration is that acting on this parameter has provided good results for some certain protocols, but it has worsened some other protocol performances. So this approach can't be used in order to enforce the system classification capability since any improvement would be coupled with some drawbacks in any other class, and this implies the decision of always analyzing only the first 5 packets, as originally done.

3.2.4 Oracle endpoint-based redefinition

It has been shown how a good training balance is compulsory in order to allow the SVM to well recognize different classes, and how a selection of a certain number of TCP packets per flow can help improving performances for some protocols. But these considerations are not sufficient to obtain a very operative classification mechanism, since the Oracle is yet port-based, so not at all reliable. In the pre-classified service traces there are yet a residuals of unknown protocols which should be put in the background class, and therefore it is important to remove them in order to obtain a clean Oracle.
The method used to redefine the Oracle is the endpoint-based filtering already described in section 3.1.3 where a pattern based Deep Packet Inspection is performed over every packets and this permits to find out which endpoints are really involved into a certain service communication.

New tests have been performed with this new traces configuration and keeping the number of analyzed packets equal to 5. The training set is always the balanced one previously described with a little modify, in fact it has been decised to try to decrease the number of *other* samples which has been set to 1800 for both cases. This should help in avoiding an excessive absorption of the service samples into the background class.

The improvement in the classification effectiveness is shown in the confusion matrix 3.7. These percentages are computed after avaraging 10 trials with random and always equally distributed training sets. Differently from the previous tests the accuracy here reaches values of 96.9% for the destination aggregation and 92.3% for the source aggregation and this so clearly results in a great enhancement.

In particular all protocols become easier to identify and this dramatically decrease the number of False Positives and the uncorrect absorption caused by the background. This problem still remains, but with a minor intensity, for *http* in both directions and, since the Oracle now is realiably instructed to recognize the true http traffic, this is caused for sure by the presence of some http connections tunnelled trough a non-standard port (e.g. port 8080) and this possibility has not been taken into consideration while filtering. To check this hypothesis all these classification errors have been verified and, also considering that traffic measurement has been

Table 3.7. Testing performances with an endpoint-based Oracle after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	94,94	0	0	0,06	2,58	0	0	0	0	0,11	0,39
https	0,04	95,57	0	0	0,32	0,78	0	0	0	0,06	2,7
ftp	0	0	98,59	0	0	0	0	0	0,03	0	0
smtp	0	0	0	99,86	0	0	0	0	0	0	0
imap2	0,02	0	0	0	90,97	0	0	0	0	0	0
imaps	0	0,05	0	0	0	89,22	0	0	0	0,19	0
skype	0,01	0	0	0	0	0	100	0	0	0	0,05
ssh	0,05	0	0	0	0	0	0	100	0	0	0,03
pop3	0,01	0	1,31	0,01	2,9	0	0	0	99,94	0	0
pop3s	0	0,2	0	0	0	10	0	0	0	98,98	0
other	4,93	4,17	0,1	0,06	3,23	0	0	0	0,03	0,65	96,82
	100	100	100	100	100	100	100	100	100	100	100

done some years ago, it has not been so strange to find only a few of active servers.

The same considerations have been done, on the SRC→DST direction, also for *https* which achieves a non-negligible False Positive rate since it is partly mistaken for background and after the same check some SSL server have been found, verifing again the hypothesis. But in this case the system has also mistaken 12132 *other* samples (the 2.7% of the total *other* objects) for *https*. This can be explained by considering again the Oracle instruction method where, as described previously, a deep inspection has not been available for this protocol since tshark doesn't provide filtering for secure protocols (SSL is recognized, but it is not able to find out the upper protocol) and so a rough 443 port filtering has been performed. So there is an amount of non-https traffic which is so labelled by the Oracle, and the system is able to understand the error.

Another problem present in both flow directions is that about the 10% of *pop3s* traffic is mistaken for *imaps*, and this can be charged to the presence of the SSL encryption in both kind of packets which is guilty of bluring partly the SVM model.

Table 3.7. Testing performances with an endpoint-based Oracle after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	91,63	0,17	0	0,07	1,54	0	0	0	0	0	13,99
https	1,02	91,27	0	0	0	0,43	0	0	0,01	0,12	1,62
ftp	0,35	6,38	98,98	0,02	0	0	0	0	0	0	1,05
smtp	0	0	0,03	99,45	0	0	0	0	0,03	0	0,03
imap2	0	0	0	0	58,08	0	0	0	0	0	0
imaps	0	0,01	0	0	0	89,78	0	0	0	0	0
skype	0,01	0	0	0	0	0	100	0	0	0	0,03
ssh	0,15	0	0	0	0	0	0	100	0	0	0,05
pop3	0	0	0	0	0	0	0	0	99,59	0	0
pop3s	0	0,65	0	0	0	9,78	0	0	0	99,88	0,02
other	6,84	1,52	0,99	0,46	40,38	0	0	0	0,38	0	83,21
	100	100	100	100	100	100	100	100	100	100	100

Finally in the DST→SRC direction there are great problems in classifying *imap2* traffic, which is absorbed by the background in percentage of 40%. This is another example of direction-dependent performances intrinsecally caused by the asymmetry of the protocol wich in a direction is easily identifiable because of the common imap2 communication dynamics, while in the other direction it is not so predictable since packets carry mainly data.

As a conclusion, in view of the evident enhancement introduced by the endpoint-based filtering mechism, all following trials exploit a so instructed Oracle.

3.2.5 Trainset dimensioning

Since a reliable enough system setup has been elaborated, it is important to conduct some more accurated test on the training set dimensioning. In fact it would be interesting to find out how an uniformally distributed set influences the classification capability, and how large should it be in order to achieve a sufficient accuracy. In fact the balanced training set used till now is composed of about 7000 samples and, if considering that the number of features to analyze is 24, it is evident how time consuming and computationally expensive would a single test be. Furthermore it should be safer to elaborate performances after 10 different trials, so computation efforts are multiplied ten times more.

Table 3.8. Testing performances with an uniformally distributed training set of 35 samples per class after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	skype	ssh	pop3	pop3s	other
http	94,67	0,02	0	0,02	0,01	0	0	0,09	3,51
https	1,18	94,82	0	0	0,07	0	0	0,18	9,34
ftp	0	0	99,31	0	0	0	0,04	0	0
smtp	0,16	0	0	99,91	0	0	0,1	0	0,01
skype	0,03	0	0	0	99,9	0	0	0	0,29
ssh	0,65	0,01	0	0	0	100	0	0	0,91
pop3	0,19	0	0,31	0,05	0	0	99,79	0	0,16
pop3s	0	3,12	0	0	0	0	0	97,2	0,08
other	3,11	2,03	0,38	0,02	0,38	0	0,07	2,53	85,69
	100	100	100	100	100	100	100	100	100

A first test has been conducted by creating a random training set composed of 35 samples per class. This number has been chosed in order to allow every classes to take part to the test with a sufficient number of testing samples: the less numerous dataset belongs to *imap2* with 70 elements, so an half is used for training. After 10 tests accuracies reached the 92% for the destination aggregation and 82% for source aggregation, proving that also with a very small training set performances achieved are good.

During a second test the training set has been raised with 100 samples per class, therefore removing *imap2* and *imaps* classes because they haven't enough samples in their datasets. Performances related to this 9-classes trial are available in table 3.8. Here accuracies reach respectively 95% and 87%, than noticeably increasing the True Positives ratio in both directions. The only "bug" stands in the background

Table 3.8. Testing performances with an uniformally distributed training set of 35 samples per class after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	skype	ssh	pop3	pop3s	other
http	84,15	0,08	0,45	0,04	0	0	0	0	19,27
https	4,01	90,29	0,02	0	0	0	0	0,26	5,54
ftp	0,95	5,78	98,66	0,68	0	0	0	0	2,86
smtp	0	0,01	0,6	99,25	0	0	0,05	0	0,18
skype	0,05	0	0	0	99,94	0	0	0	0,04
ssh	0,41	0,02	0	0	0	100	0	0	0,32
pop3	0	0,01	0	0,01	0	0	99,81	0	0
pop3s	0	2,68	0	0	0	0	0	99,72	0,2
other	10,43	1,12	0,27	0,01	0,06	0	0,13	0,01	71,58
	100	100	100	100	100	100	100	100	100

samples identification which is 10% under the other classes performances. That's clearly a drawback of the limited training set size: such a non homogeneous class can't be effectively described in only 100 samples.

3.2.6 Considering TCP flags and options

Even if the already obtained results with the current settings are more than satisfying, it has been decided to try a modify in the features set analyzed, by introducing all TCP flags and advanced options. The idea is that every protocols could use this service bits in a different way and so the analysis of their usage could be helpful in order to distinguish each single application.
To put this concept in practice a modify has been applied to the TCP-chunker module in order to produce 5 more 4bits chunks including all flags (they're 8) and TCP options (totally 12 bits). Tables 3.9 and 3.10 show respectively the considered TCP flags and options used.

Code	Name	Description
CWR	`Congestion Window Reduced`	Set by the sending host to indicate that it received a TCP segment with the ECE flag set.
ECE	`ECN-Echo`	Explicit Congestion Notification sent to reduce transmission rate.
URG	`Urgent`	Indicates the presence of urgent data at a position indicated by the offset defined by the Urgent Pointer field.
ACK	`Acknowledgment`	Indicates that the ACKnowledgment field is significant.
PSH	`Push`	Indicates that data must not be buffered but passed immediately to the upper layer.
RST	`Reset`	Reset the connection.
SYN	`Synchronization`	Connection open request and synchronization of the sequence numbers.
FIN	`Final`	Connection closing request.

Table 3.9: TCP flags list as defined in RFC 793.

Code	Name	Description
EOL	`End of Option List`	Used at the end of all options. It is only used if the end of the options would not otherwise coincide with the end of the TCP header.
NOP	`No-Operation`	A one-byte instruction which performs no operation. It's used to bound different options.
MAXSEG	`Maximum Segment Size`	Used to specify the maximum segment size that the receiver should use.

(continua)

Code	Name	Description
WS	`Window Scale`	Used to expand the definition of the TCP window to 32bits. Then a scale factor has to be used to carry this 32bits value in the 16bits Window field of the TCP header. The scale factor is carried in a new TCP option, Window Scale.
TS	`Time Stamp Option`	The timestamps are used for two distinct mechanisms: Round Trip Time Measurement and Protect Against Wrapped Sequences.
ECHO	`Echo`	This option may be sent in any segment, but only if a TCP Echo option was received in a SYN segment for the connection.
ECHOREPLY	`Echo Reply`	Used with ECHO to provide a simple method for measuring the round-trip delay.
CC	`Connection Count`	A value assigned monotonically to successive connections.
CCNEW	`Connection Count New`	Flag activated when a new connection starts and when the monotical property of CC can not be satisfied.
CCECHO		Used to echo the Current Count to validate a SYN ACK segment.
SACK_PERM	`SACK Permitted`	It may be sent in a SYN segment to indicate that the the SACK option may be used once the connection is established.
SACK	`SACK`	Used to inform the sender about all segments that have arrived successfully, allowing the sender to retransmit only the segments that have actually been lost.

Table 3.10: Advanced TCP Options list as defined in RFC793, RFC2018, RFC1072, RFC1644, RFC1323.

All new signatures are now composed of 29 χ^2 values associated to 24 payload chunks and 5 flags and options chunks. A testbed composed of 10 different trials has been perfomed and results are shown in table 3.11.

Table 3.11. Testing performances including TCP Flags and Options after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	95,19	0,05	0,61	0,14	15,48	0	0	0	0	0,22	0,45
https	0,06	97,84	0	0	0,65	1,76	0	0	0	0,03	1,28
ftp	0	0	98,59	0	0	0	0	0	0,01	0	0
smtp	0	0	0	99,78	0	0	0	0	0	0	0
imap2	0	0	0	0	80	0	0	0	0	0	0
imaps	0	0,19	0	0	0	84,71	0	0	0	0,2	0
skype	0,01	0	0	0	0	0	100	0	0	0	0,04
ssh	0	0	0	0	0	0	0	100	0	0	0,04
pop3	0	0	0,4	0,03	1,29	0	0	0	99,99	0	0,03
pop3s	0	0,2	0	0	0	13,53	0	0	0	98,93	0
other	4,74	1,73	0,4	0,06	2,58	0	0	0	0,01	0,62	98,16
	100	100	100	100	100	100	100	100	100	100	100

Some further little improvements have been achieved in terms of True Positive rates due to the greater signatures characterization, especially in the SRC→DST direction. This brought to accuracy rates respectively of 97.4% and 93% at a cost of a longer time demanded for computations, anyway this enhancement is mostly due to an increase of True Positives for the *other* class. In the source aggregation case single protocols generally see their performances degraded except for *https* which gains 1% at a cost of be partly mistaken for *http* traffic. Analyzing deeplier the destination aggregation results it is possible to find out how *http*, *https* and the *background* slightly improve their characterization. In particular https earns a 2% in True Positives and it is less mistaken for the backgound. The drawback is that *imap2* is now mistaken for *http* and looses a 10% of True Positives and *imaps* is visibly more difficult to distinguish from the other secure protocols.

A further test has been performed in order to check the service bits validity in

Table 3.11. Testing performances including TCP Flags and Options after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	https	ftp	smtp	imap2	imaps	skype	ssh	pop3	pop3s	other
http	90,89	4,53	2,72	0,05	0,38	0	0	0	0	0	6,93
https	0,33	92,11	0,13	0	6,15	0	0	0	0	0,36	0,51
ftp	0,11	1,15	96,4	0,27	0	0	0	0	0	0	0,46
smtp	0	0	0,05	98,53	0	0	0	0	0	0	0,02
imap2	0	0	0	0	57,31	0	0	0	0	0	0
imaps	0	0,34	0	0	0	83,7	0	0	0	0	0
skype	0,03	0	0	0	0	0	100	0	0	0	0
ssh	0	0	0	0	0	0	0	100	0	0	0,02
pop3	0	0	0	0	0	0	0	0	99,76	0	0
pop3s	0	0,89	0	0	0	16,3	0	0	0	99,59	0,02
other	8,65	0,97	0,7	1,15	36,15	0	0	0	0,24	0,05	92,04
	100	100	100	100	100	100	100	100	100	100	100

a traffic classification framework: a new dataset has been created forcing Tstat to produce signatures considering only the 5 chunks including flags and options and classification has been accomplished. The trial has established a total incapacity to identify services since the most part of samples were absorbed by the background, except for *http* (especially in the DST→SRC direction where it reaches a 79% of True Positives), *other* and *skype* whose performances, though not at all sufficient, showed an interesting correlation between these protocols and their bits usage. These improvements for the first two protocols have been already proved in the previous test, so for them this represents only a double-check. On the other hand this more precise characterization has not been shown for skype before since it already achieved a 100% classification True Positive rate, so these new 5 more chunks are useless in order to identify this class.

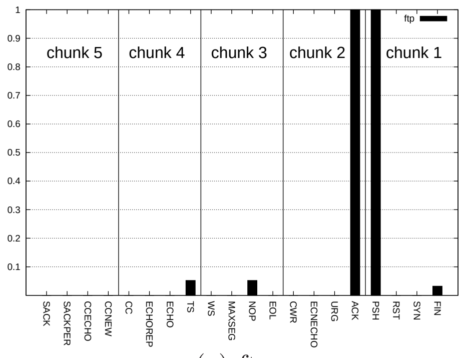
In order to better understand the dynamics related to the analysis of TCP flags and options, every service trace has been separately inspected and a counting of every single bit usage has been worked out. In this way it has been possible to infer how bits are used by each single protocol, and results of this analysis are represented in figure 3.5 which collects statistics related to all eleven considered classes. The graphs represent the bits distribution for a certain service trace defined as the ratio between the single bits usage and the number of packets seen during trace analysis. In this way it is possible to deduce relationships between a bits distribution and a protocol and reciprocal relationships among bits for a given service, which is intrinsically the identification metrics used by the SVM.

The graphs prove that bits distributions are not so characterizing each single protocols: usually TS (Time Stamp Option), NOP (No Operation), ACK and Push are set by every service with often a very similar usage distribution, while FIN is used only by some protocols but in a little significant way. Therefore only meaningful considerations are about http, showing a slightly different and unique distribution permitting it to be identified more easily and about *imap2* which in the very previous tests has been much more mistaken for http because of the similarity of their distributions.

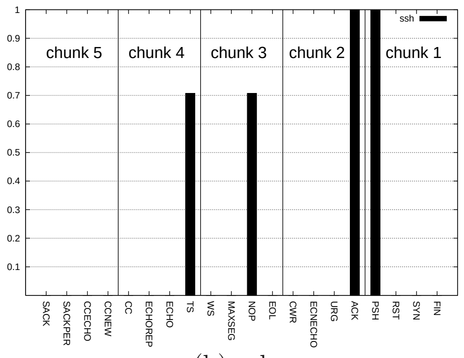
Having this cross-check shown a non remarkable benefit in the exploitation of TCP flags and options bits, their usage has been abandoned in the following tests because of high computational costs and, on the other hand, not so satisfactory improvements.

3.2.7 Getting some cleaner results

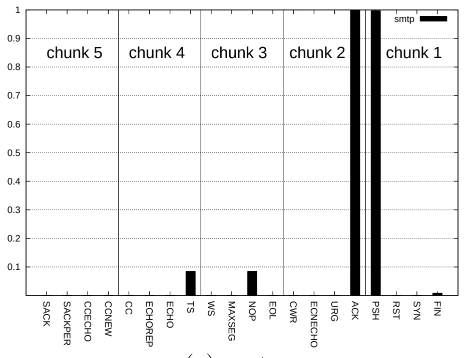
All previous tests have been carried out in order to define a satisfying system setup and so checking all possible parameters in order to maximize performances. Foremost an optimal configuration of chunking variables has been tracked down, then a balanced training set distribution has proved to provide the best results and further promising metrics have been tested. Furthermore an Oracle re-definition has been tried too, so ensuring a more reliable pre-classification and predictions validation, but, as described in section 3.1.3, also after this refining operation there were yet some classes not completely reliable, due to the absence in `tshark` of a DPI support for secure protocols. So *imaps*, *pop3s* and *https* protocols gave back the system to be not yet totally reliable. So, in order to provide clean definitive results and to test the ultimate system classification ability, these classes have been cut off from the last tests.



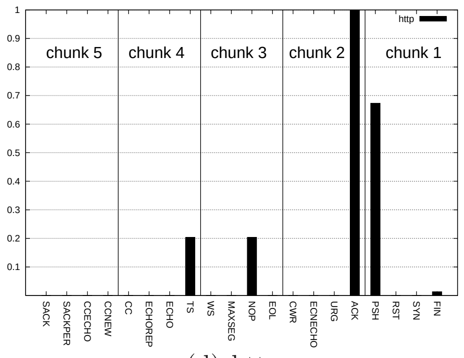
(a) ftp



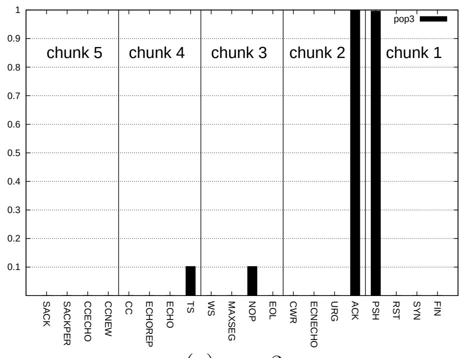
(b) ssh



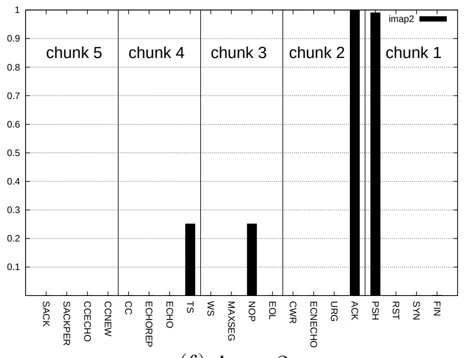
(c) smtp



(d) http



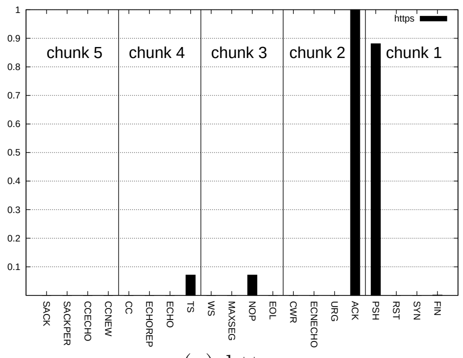
(e) pop3



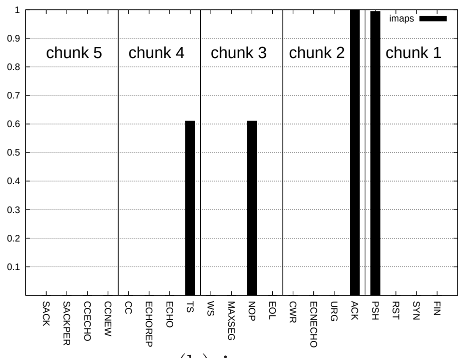
(f) imap2

Figure 3.5. TCP flags and Options distribution in the analyzed trace for each service class. The shown values represent the ratio between the number of times a certain bit has been set and the number of packets inspected. Bits are sorted on x-axis according to their actual disposition in chunks. (continue)

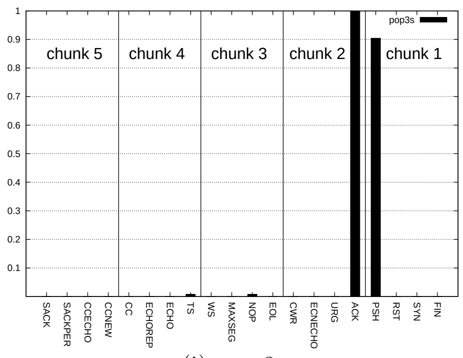
A new series of trials has been performed checking again on the best training



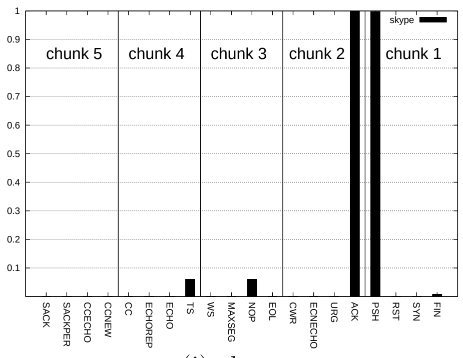
(g) https



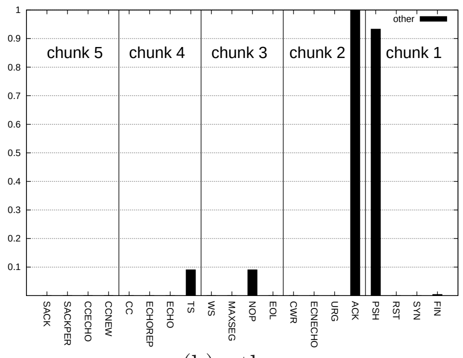
(h) imaps



(i) pop3s



(j) skype



(k) other

Figure 3.5. (continuation) TCP flags and Options distribution in the analyzed trace for each service class. The shown values represent the ratio between the number of times a certain bit has been set and the number of packets inspected. Bits are sorted on x-axis according to their actual disposition in chunks.

set to use, and one more time the balanced one has proved to be the best choise. The new classification framework is composed of a dataset split into 8 classes and,

Table 3.12. Testing performances of a reliable 9-classes system after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(a) DESTINATION

	http	ftp	smtp	skype	ssh	pop3	other
http	95,28	0,10	0,04	0	0	0	0,25
ftp	0	98,40	0	0	0	0,02	0
smtp	0	0	99,90	0	0	0,01	0
skype	0	0	0	100	0	0	0,05
ssh	0,05	0	0	0	100	0	0,02
pop3	0	1	0,02	0	0	99,91	0,01
other	4,66	0,50	0,04	0	0	0,06	99,67
	100	100	100	100	100	100	100

Table 3.12. Testing performances of a reliable 9-classes system after aggregating statistics by DESTINATION (a) and SOURCE (b) endpoints.

(b) SOURCE

	http	ftp	smtp	skype	ssh	pop3	other
http	93,31	0,11	0,01	0	0	0	7,23
ftp	0,32	99,04	0,3	0	0	0	3,07
smtp	0	0,32	99,6	0	0	0	0,01
skype	0,02	0	0	100	0	0	0,02
ssh	0,05	0	0	0	100	0	0,03
pop3	0	0	0	0	0	99,59	0
other	6,31	0,53	0,09	0	0	0,41	89,64
	100	100	100	100	100	100	100

maintaining a 24 chunks signatures configuration with a balanced training set, the very final tests have been performed and the resulting confusion matrix is provided in table 3.12. These results are the best achieved by the SVM classifier implemented and offers high accuracies respectively of 98% and 94%. The only smear concerns the *http* class since there is yet a little amount of these samples absorbed by the background and an amount of *other* samples which are uncorrectly identified as *http* (this occurs only after the source aggregation). The cause of this phenomenon is, as already seen, the presence in the background of http traffic on a non standard port causing the SVM to be trained also with some *http* samples instead of *other* ones. However these imprecisions are limited in number and don't affect too much performances, which are the best ones resulting from all trials.

Chapter 4

Conclusions

The aim of this thesis work has been to develop and test a TCP traffic classifier able to identify, through a transport layer payload statistical analysis, the application protocol used during a communication. The chosen classification metrics has been the Pearson χ^2 test computed over groups of bits extracted by the payload of packets belonging to a flow since it has formerly proven to provide excellent results in classifying objects and in particular in the applicative traffic prediction over UDP case. This statistical parameter is used as an index giving detailled informations about bits variability, so that it can be used to trace a protocol packet format by identifying different types of values such as counters, fixed or random values and mixed ones. So a Tstat chunking module has been implemented in order to produce χ^2 signatures computed periodically after a certain number of samples and they contain randomness informations about every groups of bits extracted by each flow's packets payload, by aggregating statistics in order to consider every endpoints separately. Such an aggregation can provide informations separately about both flow directions, allowing to conduct direction based analysis and to deduce different classification performances related to each protocol dynamics Signatures produced have represented the discrimination features for traffic classification, put into practice by means of a Support Vector Machine which maps the χ^2 vectors as points into a higher dimensional space providing high prediction capabilities.
A meticolous testing has shown how a correct and reliable SVM learning phase is decisive in order to obtain good performances and in particular special care must be given to the training set composition and to a clean pre-filtering operation in

view of a correct system instruction and predictions validation. These aspects have been managed by composing a balanced training set where samples are distributed in a progressive fashion such that every class is described by a sufficient number of objects and no class is oversampled. Furthermore, an effective pre-filtering mechanism based on a exploitation od Deep Packet Inspection and port/endpoint based techniques has been thought in order to feed the SVM with pure splitted applicative traffic. The only problem which can be corrected afterward is represented by the erroneous presence in the background subset of some HTTP tunnelled traffic trough a non-standard port, slightly degrading HTTP identification performances. Some other approach have been put to trial, such as a study of the number of packets per flow to analyze in order to improve performances, or the idea to consider also TCP flags and advanced options contained into the TCP layer header. However results have improved not so significantly with the drawback of an huge increase in computational costs, so that these approaches have been missed out.

The best system setting performances achieved corresponded to accuracies of 98% for the destination aggregation case and 94% for the source aggregation case, showing a very satisfying result evaluation for the implemented system with a low error rate. Since these testing have been applied only to an old traffic trace, it should be interesting to update the system to be ready to analyze real time traffic too. It must be noted also that this classifier can't produce prediction after each packet analyzed, but it must wait for a certain number of samples to produce statistical samples to be classified. This feature corresponds to a non-instantaneous classification capability which can represent an drawback intrisic to the statistical method implemented.

Bibliography

[1] D. Bonfiglio, A. Finamore, M. Mellia, M. Meo, D. Rossi, *KISS: Chi-Square Signatures for Internet Traffic Classification*.

[2] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, Kave Salamatian, *Traffic Classification On The Fly*, ACM SIGCOMM Computer Communication Review, Volume 36, Number 2, April 2006.

[3] Jeffrey Erman, Martin Arlitt, Anirban Mahanti, *Traffic Classification Using Clustering Algorithms*, SIGCOMM 06 Workshops September 2006, Pisa, Italy.

[4] Marco Mellia, Renato Lo Cigno, Fabio Neri, *Measuring IP and TCP behavior on edge nodes with Tstat*, Computer Networks, Vol.47, No.1, pp.1-21, ISSN: 1389-1286, Jan 2005.

[5] W. Luo, X. Liu, J. Zhang, *SVM-based analysis and prediction on network traffic*.

[6] Christopher J.C. Burger, *A Tutorial on Support Vector Machines for Pattern Recognition*, Kluwer Academic Publishers, Boston.

[7] Anukool Lakhina, Mark Crovella, Christophe Diot, *Mining Anomalies Using Traffic Feature Distributions*, SIGCOMM 05, Philadelphia, Pennsylvania, USA.

[8] C. Hsu, C. Chang, C. Lin, *A Practical Guide to Support Vector Classification*, Taipei, May 21, 2008.

[9] A. Finamore, *Applicazione di metodi statistici alla classificazione di traffico in reti dati*, Torino 2008.

[10] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a Library for Support Vector Machines*, 2007.

[11] Vladimir N. Vapnik, *The nature of Statistical Learning Theory*, 1995.

[12] Vladimir N. Vapnik, *An Overview of Statistical Learning Theory*, IEEE Transactions on neural networks, VOL. 10, No. 5, September 1999.

[13] Carl Staelin, *Parameter selection for support vector machines*, HP Laboratories Israel, November 10, 2003.

[14] C. Cortes and V. Vapnik, *Support-vector network. Machine Learning*, 1995.

[15] B. Schölkopf, A. J. Smola, *Learning with kernels: Support Vector Machines, regularization, optimization, and beyond*, MIT Press, 2002.

[16] J. Milgram, M. Cheriet, R. Sabourin, *"One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?*

[17] S. Keerthi, C. Lin, *Asymptotic behaviours of Support Vector Machine with Gaussian kernel*, 2003.

[18] H. Lin, C. Lin, *A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods*, 2003.

[19] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, Paolo Tofanelli, *Revealing skype traffic: when randomness plays with you*, ACM SIGCOMM Computer Communication Review "4", Vol.37, pp.37-48, ISSN: 0146-4833, October 2007.

[20] Tstat website, `http://tstat.tlc.polito.it`

[21] LibSVM website, `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[22] Python reference website, `http://docs.python.org/index.html`

[23] Bash scripting reference website, `http://www.gnu.org/software/bash/bash.html`