# Mining Unclassified Traffic Using Automatic Clustering Techniques

Alessandro Finamore, Marco Mellia, and Michela Meo

Politecnico di Torino
lastname@tlc.polito.it

**Abstract.** In this paper we present a fully unsupervised algorithm to identify classes of traffic inside an aggregate. The algorithm leverages on the K-means clustering algorithm, augmented with a mechanism to automatically determine the number of traffic clusters. The signatures used for clustering are statistical representations of the application layer protocols.

The proposed technique is extensively tested considering UDP traffic traces collected from operative networks. Performance tests show that it can clusterize the traffic in few tens of pure clusters, achieving an accuracy above 95%. Results are promising and suggest that the proposed approach might effectively be used for automatic traffic monitoring, e.g., to identify the birth of new applications and protocols, or the presence of anomalous or unexpected traffic.

## 1 Introduction

The identification and characterization of network traffic is one of the most important activities for an operator. Through the continuous monitoring of the traffic, security policies can be deployed and tuned, anomalies can be detected, changes in the users behavior can be identified so that QoS and traffic engineering policies can be continuously improved.

In the last years, several traffic classification techniques have been proposed. At the beginning *port-based* approaches were mainly used; however, the characteristics of many nowadays applications that employ randomly chosen ports, significantly reduce the effectiveness of these approaches [1–4]. Those are today abandoned in favor of *deep packet inspection* (DPI) or *behavioral* techniques [13, 15]. In the first case, the traffic is classified looking for specific keywords inside the packet payload, e.g., `BitTorrent` or `GET/POST` keywords identify the BitTorrent and HTTP protocols, respectively. Behavioral techniques try to overcome the limitations of DPI, e.g., when payload is encrypted, by exploiting some description of the application behavior through statistical characteristics, such as the length of the first packets of a flows.

All these classifiers share some key aspects. On the one hand a deep domain knowledge is required to correctly train and periodically update these classifiers. On the other hand, the classifiers can identify only the specific applications they have been trained for; all other traffic is aggregated in a single class labeled

J. Domingo-Pascual, Y. Shavitt, and S. Uhlig (Eds.): TMA 2011, LNCS 6613, pp. 150–163, 2011.

as "unclassified". The classifiers are therefore typically tuned to identify the prominent classes but they completely miss the dynamics of the rest of the traffic. For example, they cannot identify the introduction of a new application, or changes in the users' behavior or in the applications protocols.

Classification can happen at different degrees of granularity: *packet*, *flow*, or *endpoint*[1], with significant differences on the number of objects to be considered. However, when mining the subset of unclassified traffic, the number of objects to be analyzed is still large even when considering higher aggregation levels. For instance, for moderate traffic aggregates, the even small fraction of unclassified traffic is typically built by thousands of endpoints, each aggregating tens of flows made of hundreds of packets. How to practically reduce the number of unknown objects to analyze is therefore a key problem.

In this paper, we focus our attention on the inspection of the unclassified traffic. We propose an unsupervised technique that, having no knowledge of the applications that generate the traffic, partitions a traffic aggregate into "clusters" that are distinguished based on common features, i.e., they exhibit a common treat. A simple clustering methodology based on the K-means algorithm is augmented with the capability to effectively determine the number of traffic clusters $K$. The results is a simple algorithm that can reduce the number of objects to analyze to few tens, even if the total traffic amounts to several tens of megabits per seconds. By being completely automatic and unsupervised, the proposed methodology can be engineered to: i) identify new classes of traffic by exploiting the network administrator domain knowledge when inspecting a traffic cluster; ii) monitor the traffic evolution by highlighting the birth of traffic clusters corresponding to traffic of previously unobserved applications; iii) design anomalies detection techniques by observing the evolution of traffic clusters over time.

To test and validate our methodology, we consider some UDP traffic traces of which we already have a deep knowledge on, achieved through a combination of DPI and statistical techniques, as well as the results of some active experiments. We consider UDP traffic since today its importance is steadily increasing [4], and few works explicitly targeted it in the past. We apply the proposed technique to the traces and check the coherence of the automatic classification with our ground truth. Experimental results show that the proposed clustering algorithm is very effective. Clusters accuracy is typically higher than 95% and the number of clusters is also very small, e.g., never larger than 40, and typically in the order of 25. Such a good performance is due to both the descriptiveness of the KISS features, and the goodness of the agglomerative process. With respect to previous proposals [5, 6] in which hundreds of clusters were needed to achieve good accuracy, the major advantage of our solution is that it reduces the time needed to inspect the clusters since the traffic is better partitioned.
Finally, we present some examples of classification of unknown traffic we were able to identify.

[1] A flow is commonly defined as the group of packets that have the same tuple {srcIP, dstIP, srcPort, dstPort, protocol}. An endpoint identifies the group of flows having the same {host IP, host Port, protocol} tuple.

## 2 Data Mining Techniques and Related Work

Machine learning algorithms are data mining techniques used to create a model from a dataset. They can be grouped in two families: *supervised* and *unsupervised* techniques. In both cases, objects are characterized by features, i.e., a vector of characteristics that can be extracted automatically by observation. Supervised algorithms exploit a training dataset in which each object is labeled, i.e., it is a-priori associated to a particular class. This coupling is used to create a suitable model so that objects with the same labels are grouped together. Then, unlabeled objects can be associated to a class previously defined according to their features. For unsupervised algorithms, instead, the grouping operation is automated without any knowledge of a-priori labels. Groups of objects are then clustered based only on a notion of distance evaluated among samples, so that objects with similar features are part of the same cluster. Supervised algorithms allow high accuracy during classification, provided that the training set is representative of the objects.

The application of machine learning techniques is not new in the traffic classification field. [7] is one of the preliminary works and shows that clustering techniques are useful to obtain insights about the traffic. In [6] supervised and unsupervised techniques are compared, demonstrating that unsupervised algorithms can achieve the same performance of the supervised algorithms. Other works compare the accuracy of different unsupervised algorithms [3, 5, 8]. In general, the techniques presented in these works achieve a very high accuracy but they typically require several hundreds of clusters, therefore making it difficult to then inspect and label the clusters. Recently, [9] and [10] have introduced the *semi-supervised* methodology. They exploit the advantages of both methodologies: a clustering algorithm is used to partition the dataset as in the unsupervised case. Part of the dataset is labeled, so that it is possible to extend the classification to all objects in the same cluster. Results shows that the accuracy of the classification largely depends on the goodness and coverage of the labeled dataset, and clusters without labeled objects cannot be further classified.

All previous works focus on the classification accuracy of some target classes, i.e., a small subset of the applications to consider. Real traffic is however composed by a large mix of applications and often it is crucial to mine the remaining part of the traffic which is still unclassified. For example, in [3] authors show that the best classifier has poor performance when considering the unclassified traffic which amounts to more than 10% of the total.

In addition, the Internet represents a dynamic environment in which new applications are born, evolve and die continuously. By following these patterns, it is possible to better understand the users behaviors and the technology trends.

## 3 Feature Selection: Kiss Signatures

Machine learning algorithms are based on a description of objects summarized in a vector. The elements of the vector are called *features* and constitute a description of all known characteristics of the instance. They play a key role in the

effectiveness of the machine learning algorithm, i.e., the more descriptive the features are, the better the performance is. In the past, most of the works considered a large set of generic features, such as packet/flow length, port number, round trip time. In this paper, instead, we rely on the signatures defined by Kiss, a stochastic classifier that we proposed in [11, 12]. The intuition behind Kiss is that application-layer protocols can be identified by statistically characterizing the stream of bytes observed in a flow of packets. Kiss builds protocol signatures by measuring the randomness of groups of bits extracted from the packets payload. Considering an analogy, this process is like recognizing the foreign language by considering only the cacophony of the conversation, i.e., by letting the protocol format emerge, while discarding its actual semantic. Kiss features proved to be highly descriptive when adopted in supervised machine learning algorithms for traffic classification.

Kiss signatures are computed over the packets directed to or originated from a given *endpoint*. They aim at measuring the randomness of the first bytes of the packet payload that are those usually carrying application header. In particular, the first 12 bytes of the packet payload are divided into groups of $b = 4$ bits, for a total of $G = 24$ groups. For each group, the statistic of the occurrence of each of the $2^b = 16$ possible values is computed over $N = 80$ packets. Then, the randomness of each group $g$, denoted by $X_g$, is measured as the Chi-Square distance of the group statistics with respect to the uniform distribution,

$$X_g = \sum_{i=0}^{2^b-1} \frac{(O_i^g - E_i)^2}{E_i} \tag{1}$$

where $O_i^g$ is the observed occurrence of the value $i$ for the $g$ group, and $E_i = N/2^b$ is the expected occurrence for the uniform distribution. Finally, since the value of $X_g$ grows exponentially with the number of deterministic bits in the group, and linearly with $N$ [11], we derive,

$$b_g = \log_2 \left( \frac{X_g}{N} + 1 \right) \tag{2}$$

where $b_g$ represents then the number of constant bits in group $g$. The vector $\{b_1, b_2, \ldots, b_G\}$ represents the Kiss signature used in the rest of the paper.

Since Kiss features are obtained from inspection of packet payload, encrypted application layer protocols may limit the goodness of the features, i.e., all groups may look like random data. In those cases, it would results impossible to correctly distinguish two applications that adopt fully encrypted payload.

In summary, each Kiss signature computed from the traffic of an endpoint corresponds to a "point" in an hyper-space of 24 dimensions. Given then a set of monitored endpoints and of corresponding Kiss signatures, the objective of this work is to identify "clouds" of similar points, i.e., to clusterize the signatures with no a-priori knowledge about the applications that generated the traffic. Resulting clusters are higher level objects that can be investigated further, and whose properties naturally extend to each endpoint in it. For example, the clusters

give indications about how the traffic volume distributes among the regions suggesting which dataset should be further inspected. Similarly, by constantly monitoring the galaxy of clouds in time, it would also be possible to identify traffic shifts due to the rise/decline of applications, to the presence of anomalous behaviors, or malicious users.

## 4 Clustering Methodology

Kiss signatures map the traffic generated by applications into points in an hyper-space. To partition the space into pure clusters where points are generated by the same application, we leverage on the *K-means* algorithm, a classic unsupervised technique [13]. Given a set of $K$ "centroids", the K-means algorithm iterates over two steps: it first assigns each point to the closest centroid, defining a cluster; then, each cluster centroid is re-computed as the arithmetic mean among all points of the cluster. The algorithm ends either after a predefined number of iterations or if centroids do not change at a given iteration. At the beginning, centroids are randomly picked.

The major drawback of K-means is that it assumes the a-priori knowledge of the number $K$ of clusters one is interested in. The proposed algorithm tries to overcome this limitation using an agglomerative approach. We start by decomposing the hyper-space in a large number of clusters, $K_0$. Then, we incrementally merge the two closest clusters until one cluster only remains. A similar technique was successfully applied to the network measurement context in [14]. The pseudo-code of the algorithm is:

```
K = K0
centroids, labels = K-means(K, data)
while (K > 1)
    c1, c2 = closest_centroids(centroids)
    centroids = merge_centroids(centroids, c1, c2)
    labels = redo_labeling(data, centroids)
    K = K - 1
```

We start running K-means with $K_0 = 100$ randomly chosen centroids, i.e., we force the partitioning of the hyper-space in a large number of small clusters that are extremely pure. The algorithm then iterates merging at each step the two closest clusters: at step $K$, the algorithm looks for the two closest centroids $c1, c2$, it merges them into a new centroid positioned at the geometric barycenter of $c1$ and $c2$; then points are reassigned to the new set of $K - 1$ centroids. The algorithm continue the aggregation until 2 clusters only remain.

The rationale behind the algorithm is that two centroids which are very close are likely to be associated to the same final cluster. By monitoring the value of the closest distance between centroids at each iteration step, and using this as an *indicator function*, it is possible to decide the optimal value of $K$, namely $K_c$, to stop at.

In our scenario, $K_c$ represents the estimated number of protocols that are present in the dataset. Let the smallest distance between centroids be defined as

$$\gamma_K = (d_K - d_{K-1})^2 \tag{3}$$

where $d_K$ is the Euclidean distance between the two closest centroids at step $K$ of the algorithm. $\gamma_K$ defines our indicator function. Since the distance between points (and clusters) that correspond to the same protocol is expected to be smaller than the distance between points that correspond to traffic generated by different applications, large values of $\gamma_K$ suggest that the algorithm is artificially enforcing the merging of two clusters that are quite different from each other.

Notice that only a single run of K-means is executed at the beginning to obtain the initial set of clusters. At each iteration, the algorithm works only on the centroids, and this has two main benefits. First, we can better control the modification on the space due to aggregation. In fact, the K-means algorithm is subject to "oscillation effect", i.e., small modifications in the centroid position could lead to large transformations in the cluster geometry. By using centroids only we avoid to re-assign samples to centroids, so that the quality of the initial clustering is better preserved. In addition, by considering centroids only we reduce the computational cost by several order of magnitudes, we handle $O(K_0)$ centroids instead of $O(N)$ samples ($N >> K_0$). Moreover, the K-means complexity depends on the maximum number of iterations $I$ (which in our case we set to 100), so that its complexity is $O(IN)$. In our experiments on an AMD Athlon-64 X2 Dual Core Processor 4200+, we elaborated several thousands of points present in a 15 minute long traffic traces in less than 3 minutes, the largest majority of the time being devoted to the initial K-Means run. Given that the code used can be further optimized, the result is promising and suggests that the algorithm might be applied to real-time monitoring.

Finally, K-means is known to suffer from the choice of the initial centroid position. Usually initial centroids are randomly chosen so that different starting conditions can lead to different clustering. In our scenario, since we select a large number $K_0$ of clusters, the bias introduced by the selection of the initial centroids is minimal. We performed some tests by running the algorithm with different initial random seeds and the results (not reported for the lack of space) show that there is practically no influence on the initial choice.

## 5 Experimental Results

### 5.1 Datasets

The results presented in this paper refer to datasets extracted from two traces, called *ISP-Trace* and *P2PTV-Trace*; the traces are described in Table 1.

ISP-Trace is a real traffic trace collected from the network of an Italian large ISP which offers converged services, in which data, native VoIP, and IPTV share a single broadband connection. This dataset is representative of a very heterogeneous scenario, in which users are free to use the network without any restriction.

**Table 1.** Description of the ISP-Trace (a) and P2PTV-Trace (b)

| | Protocol | #flows $\times 10^3$ | (%) | Mbytes | (%) | #endp. $\times 10^3$ | (%) | #sign. $\times 10^3$ | (%) |
|---|---|---|---|---|---|---|---|---|---|
| (a) | BitTorrent | 217 | (3.39) | 40 | (0.19) | 34 | (4.14) | 22 | (0.33) |
| | DNS | 260 | (4.05) | 185 | (0.88) | 153 | (18.79) | 31 | (0.47) |
| | eMule | 5200 | (80.96) | 936 | (4.43) | 476 | (58.56) | 61 | (0.91) |
| | RTCP | 8 | (0.13) | 46 | (0.22) | 6 | (0.73) | 25 | (0.38) |
| | RTP | 9 | (0.14) | 18244 | (86.26) | 7 | (0.86) | 6222 | (92.14) |
| | Unclassified | 728 | (11.34) | 1698 | (8.03) | 137 | (16.92) | 390 | (5.78) |
| | *tot* | 6422 | (100.00) | 21149 | (100.00) | 813 | (100.00) | 6751 | (100.00) |
| (b) | PPLive | 27 | (78.52) | 1585 | (32.96) | 184 | (38.90) | 23 | (28.30) |
| | SopCast | 5 | (14.87) | 2282 | (47.43) | 176 | (37.21) | 48 | (57.46) |
| | TVants | 2 | (6.61) | 943 | (19.61) | 113 | (23.89) | 12 | (14.24) |
| | *tot* | 34359 | (100.00) | 4810 | (100.00) | 473 | (100.00) | 83 | (100.00) |

It therefore is a very challenging scenario for traffic classification. In this paper we present results considering a dataset obtained monitoring a PoP for 24 hours in October 2007, during which about 21GB of UDP traffic and 813,000 end-points were monitored. Some *known* protocols (BitTorrent, eMule, RTP, RTCP and DNS) have been extracted from the aggregated trace using Tstat [15], a traffic classifier that combines a number of DPI mechanisms with statistical techniques. The classification has been manually cross-checked to have a high confidence in the ground truth. These protocols account for more than 90% of the total volume, as shown in Table 1. The remaining 10% of traffic has been labeled as "unclassified".

P2PTV-Trace was collected during ad-hoc experiments that were organized to observe the performance of popular P2P-TV applications, namely PPLive, SopCast and TVants. The resulting dataset [16] consists of packet level traces collected from more than 45 PCs running P2P-TV applications in 5 different Countries, and it is representative of a wide range of different scenarios. Being the result of active experiments, the trace contains only a single protocol at a time and we have a perfect knowledge about it.

The datasets extracted from the two traces are disjoint. In fact, there is no P2P-TV traffic in the ISP-Trace. When needed, we can artificially "inject" P2P-TV traffic from the P2PTV-Trace into the ISP-Trace to increase the number of known protocols when assessing the performance of the clustering algorithm.

### 5.2 Evaluation of the Proposed Approach

Fig. 1(a) shows the evolution of the indicator function during the application of the algorithm to ISP-Trace considering a 10 minute long trace. The minimum distance between clusters is very small for values of $K > 20$, suggesting that the algorithm is merging clusters whose centroids are very close. Instead, for $K \leq 20$, the algorithm starts merging cluster centroids which are quite far from each other, suggesting an improper and artificial merging.

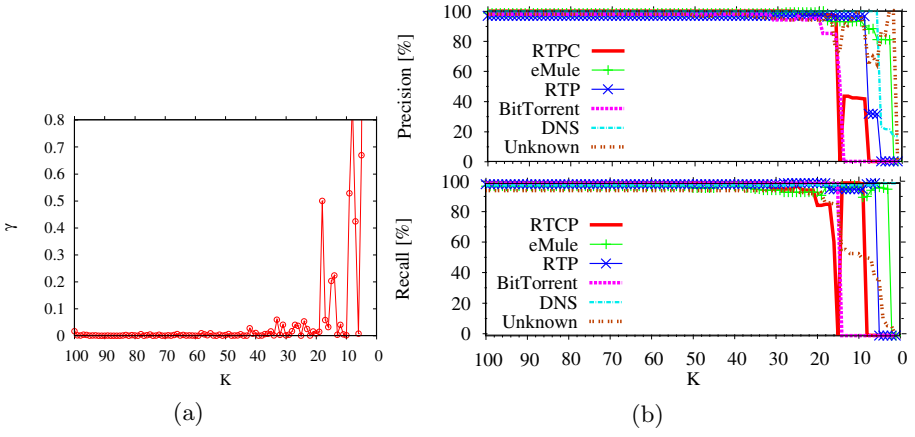



**Fig. 1.** Evolution of the clustering algorithm: the indicator function (a) and classification accuracy in terms of precision and recall (b)

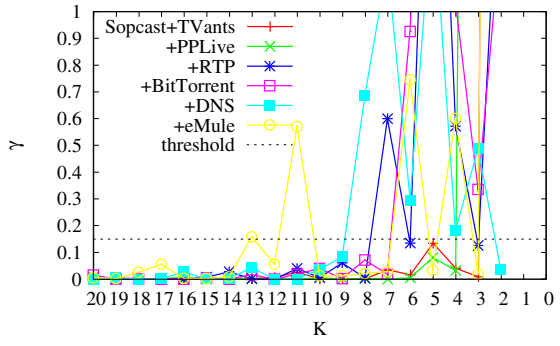
To confirm this intuition, the homogeneity of each cluster is evaluated against the endpoint classification obtained by Tstat (our ground truth). Fig. 1(b) reports the precision (top) and recall (bottom) performance indexes, defined as

$$Precision = \frac{true\ pos}{true\ pos + false\ pos} \qquad Recall = \frac{true\ pos}{true\ pos + false\ neg} \tag{4}$$

for different values of $K$. Precision is a measure of exactness or fidelity, whereas recall is a measure of completeness; these two measures complement each other. A precision of 1.0 for a class C means that every item labeled as belonging to C does indeed belong to C. It however says nothing about the number of items from class C that were not labeled correctly. A recall of 1.0 means that every item from class C was labeled as belonging to class C. It however says nothing about how many other items were incorrectly labeled as belonging to class C.

Consider Fig. 1(b); two observations hold. First, for $K > 20$, the fidelity and completeness of the identified clusters is very high, proving that the Kiss signatures accurately represent different protocols, and that traffic generated by different applications can be easily clustered. Second, the abrupt decrease of both precision and recall observed in Fig. 1(b) for $K \leq 20$ confirms that some clusters corresponding to different protocols are artificially merged, causing the formation of impure clusters.

To further assess the goodness of the approach, we inspect the behavior of the indicator function considering datasets in which we progressively add traffic of various applications. We start by considering a dataset containing only SopCast and TVants traffic; we then add, in sequence, the traffic of PPLive, RTP, BitTorrent, DNS and eMule to the dataset. For each traffic mix we run our algorithm. The results are reported in Fig. 2 for $K \leq 20$, only. The figure shows that the indicator function abruptly increases for values of $K$ that are strongly related with the number of traffic classes. A simple thresholding mechanism on




| Classes | $K_c$ | Precision | Recall |
|---|---|---|---|
| 2 | 3 | 100 | 100 |
| 3 | 4 | 100 | 100 |
| 4 | 8 | 100 | 100 |
| 5 | 7 | 99.24 | 99.2 |
| 6 | 9 | 98.13 | 97.88 |
| 7 | 14 | 98.4 | 98.18 |

**Fig. 2.** Example of indicator function for traffic aggregates with progressively increasing number of classes

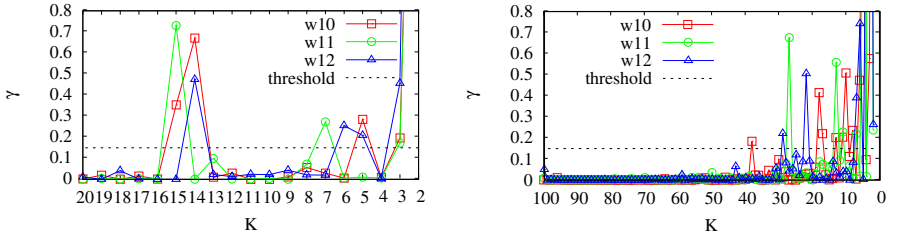



**Fig. 3.** Evolution of the clustering algorithm over different time windows of the ISP-Trace without (a) and with (b) the unclassified traffic

the indicator function can be adopted to automatically detect the value $K_c$. As an example, the figure reports a threshold of 0.15 that resulted very effective in our tests.

The Table on the right of Fig. 2 reports the suggested number of clusters $K_c$ obtained with the threshold $\gamma = 0.15$, the corresponding recall and precision are also indicated. Results confirm that the value of $K_c$ increases with the number of traffic classes. The resulting precision and recall are extremely high, and a marginal decrease is observed only when considering more than 5 protocols. This is due to BitTorrent traffic which is sometimes confused with TVants traffic whose Kiss signatures result similar. Nevertheless, the performance are very good.

Interestingly, the number of identified clusters is larger than the actual number of applications. This is due to single applications using multiple protocols with different formats, e.g., signaling is different respect to data messages. The Kiss signatures are therefore different, and the clustering algorithm correctly identifies separate clusters.

Finally, we repeat the experiment considering other different 10-min-long traces extracted from the ISP-Trace. The goal is to investigate if the indicator function always correctly suggests the number of cluster to use. Fig. 3(a)

reports the indication functions obtained for the three windows considering the aggregation of the 7 considered traffic classes. No unclassified traffic is present. We can see that the suggested $K_c$ is consistent among all the experiments. Instead, Fig. 3(b) reports the indicator functions when unclassified traffic is present too. In this case, since different traffic mixes are present during different periods of time, higher noise is present with respect to the previous case and different values $K_c$ are selected in different windows. In conclusion, the indicator function suggests an optimal number of clusters which changes depending on the actual traffic mix. Thus, a conservative large number of clusters is preferable, especially when considering different time windows. Moreover, note that in Fig. 3(b) the number of suggested clusters is never higher than 40.

### 5.3 Comparison with Other Clustering Techniques

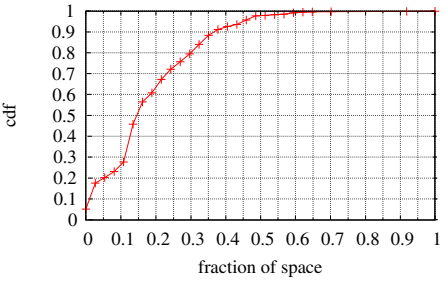
The automated selection of the optimal number of clusters is not new in literature. Several score indexes have been proposed to precisely correlate the goodness of the clustering with the number of used clusters. Examples of these indexes are: the Bayesian Information Criterion (BIC) adopted by the XMeans algorithm [13] and the Normalized Mutual Information (NMI) [3]. In this work, we are interested in investigating the automated approaches which do not require the a-priori knowledge of the points' labels (that is instead required by the NMI). We evaluated the performance of both XMeans and NMI; in addition, we considered also the DBScan algorithm. XMeans shows similar performance as our algorithm in terms of recall and accuracy. However, the number of identified clusters is typically much larger than the one obtained by our algorithm. For example, XMeans accuracy is higher than 95%, but at least 10 more clusters are identified, i.e., 50% more than with our proposal. Considering NMI, the accuracy is lower than 95% when 25 clusters are used, as suggested by the NMI technique. With 40 clusters, performance of the NMI-based method is similar to the one of our algorithm. Finally, DBScan performed poorly achieving only 85% of accuracy with the best parameter setting.

Notice also that all previous algorithms are computationally more expensive than our proposal. In conclusion, the proposed algorithm is completely automated, does not require any knowledge of the points labels and seems a good trade-off among clustering accuracy, number of clusters and complexity.

### 5.4 Clusters Distances

In this section, we investigate the geometry of the clusters of points identified by our algorithm. The results presented in this section are obtained using $K = 40$ (a conservative large value) and refer to a single time window of ISP-Trace. For the other time windows, not shown here for the sake of brevity, we obtained similar results.

We start by considering the size of each cluster. Fig. 4 shows the cumulative distribution function of the normalized Euclidean distance between each point and its centroid. As we can see, half of the points in the dataset are very close to




| Class | #cluster | #small | #not-dense |
|---|---|---|---|
| BitTorrent | 1 | 1 | - |
| RTP | 1 | 1 | - |
| DNS | 3 | 1 | - |
| RTCP | 7 | 2 | 4 |
| eMule | 10 | 6 | 7 |
| Unclassified | 22 | 20 | 18 |

*#small*: n.o. clusters with radius $\leq 0.2$
*#not-dense*: n.o. clusters with $\leq 10$ points

**Fig. 4.** Cumulative distribution function of the distance between points and centroid in each cluster (plot on the left) and a few additional data (table on the right), obtained running the algorithm on ISP-Trace with $K = 40$

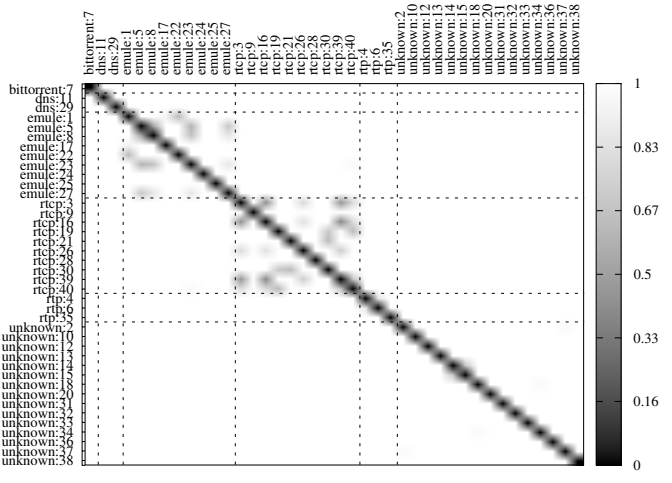



**Fig. 5.** Distance between centroids of different clusters, for 40 clusters obtained by running the algorithm on IPS-Trace with $K = 40$

theirs centroid, with a distance smaller than 20% of the cluster space size. The table on the right of Fig. 4 reports some statistics about the clusters geometry according to the DPI classification. In particular, the second column reports the number of clusters identified for each application, the third column reports the number of *small* clusters, i.e., clusters with a radius smaller that 0.2, and the last column gives the number of *not-dense* clusters, i.e., clusters with less than 10 samples. BitTorrent and RTP are mapped into a single cluster, while the "unclassified" is composed of a set of small, often not-dense, clusters. Interestingly, eMule is highly partitioned too. Investigating further, we noticed that each cluster corresponds to a different protocol which eMule uses for different purposes, e.g., one protocol is used to exchange messages with the server, another one is used to exchange traffic with peers.

To better understand the possible overlapping between the clusters, Fig. 5 reports the distance between pairs of centroids. The distance has been mapped

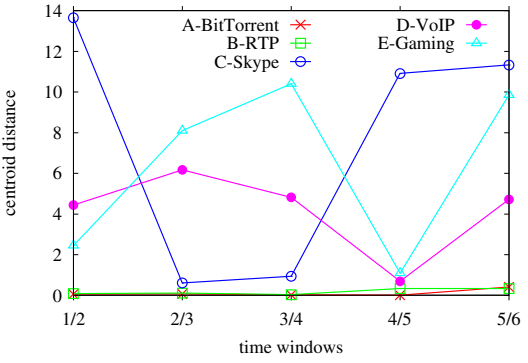



**Fig. 6.** Example of evolution of the centroids position

onto a gray scale map in which the darker the color is, the nearer the centroids are. The image is symmetrical with respect to the main diagonal, where all points have a distance of 0 by definition. Clusters are ordered based on their type of traffic so that clusters referring to the same application are nearby; dashed lines are used to delimit the applications. The only blocks which include nearby clusters are related to the same application.

In conclusion, we can say that the Kiss signatures map different protocols in different compact clusters of the hyper-space. The geometry of the clouds is strictly related to the characteristics of the application, but the signatures are naturally clustered in pure areas which do not overlap.

## 6 Mining the Unclassified Traffic

In this section we show how the proposed technique can be used to monitor the traffic evolution in time and detect the presence or absence of traffic in different periods. To do so, we measure the modifications of the clouds obtained by running the algorithm over consecutive time windows. We consider 1 hours of traffic divided into six 10-min-long traces and for each trace we run our algorithm using $K = 40$, as previously described. The centroids obtained for each time window are then compared with centroids identified in the previous time window. Each centroid is associated to the closest cluster in the previous set according to their geometric distance. This allows to detect changes between the current and the previous cluster placement.

Fig. 6 reports some interesting examples; it shows the distance of some selected centroids in consecutive time windows. For example, the position of centroid A and centroid B is practically the same over time. Verifying the corresponding clusters, we found out that samples of cluster A and cluster B are associated to BitTorrent and RTP, respectively; since in the traffic traces those applications are always present, the corresponding centroids are always present and more or less in the same position.

Consider now the case of the cluster with centroid C. The minimum distance among the centroid C in the first and the second time window is very high,

suggesting that in the second time window C is associated to a cluster of traffic that was not present during the previous time window. When comparing centroid C to its closest centroid at time window 3, we see that it moved very little. Similarly, considering time windows 3 and 4, centroid C is still referring to the same cloud of samples. Only in time window 5, the centroid C seems to disappear, since the closest centroid is very far from its position during time window 4. This suggests that some new traffic appears at the 2-nd time window, it is present during the 3-rd and 4-th time window, when it disappears again. Investigating further, we discovered that the traffic was generated by a Skype call that lasted for that period of traffic. Centroid C then refers to Skype Voice protocol.

Similar conclusions can be drawn following centroid D and centroid E evolution. Comparing their position during the 4-th and the 5-th window, we can observe that they moved little, i.e., they refer to the same cluster. Manual inspection revealed that the traffic of cluster-D corresponds to STUN protocol - Simple Traversal of User Datagram Protocol that was initiated by some P2P client that was alive in time window 4 and 5. Centroid E refers, instead, to traffic between hosts that used port 16567. This latter is composed by both short packets and much bigger packets, which might be related to Battlefield2 protocol.

Beside these examples, the methodology identified other sets of clusters and centroids which were always placed in the same zone across consecutive the windows. Some of these clusters were due to long-lived, single connections carrying many bytes, while others contained P2P-like flows, i.e. endpoints exchanging limited amount of data with an large number of hosts. Unfortunately, because of the limited amount of available payload, we are not able to further identify the application that generated these flows.

These examples show how we could successfully employ our technique to get insights into the unclassified traffic that Tstat DPI and behavioral classifiers cannot identify. In terms of traffic volumes, we could correctly identify and clusterize more than the 40% of unclassified traffic.

## 7 Conclusion

In this paper, we presented a clustering methodology to partition a traffic aggregate in classes according to the generating application. Using statistical signatures as those of Kiss, one of our classifiers, the methodology (that is completely unsupervised) is based on the K-Mean clustering algorithm enhanced through a mechanism to detect the optimal number of clusters.

Results show that the traffic partitions are very accurate. and confirm that the statistical signatures are effective in capturing the differences among application protocols. Moreover, our results prove that the methodology can be effectively used in different contexts. First of all, it is helpful to mine the unclassified traffic, i.e. the traffic that traditional DPI or a behavioral classifiers cannot recognize. Indeed, it helped us revealing 40% of the traffic we could not classify with our classifiers. Second, the algorithm can reveal the born of new applications, as well as the changes of existing ones.

## References


1. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is p2p dying or just hiding? In: Globecom, Dallas, TX (November 2004)
2. Karagiannis, T., Papagiannaki, D., Faloutsos, M.: Blinc: Multilevel traffic classification in the dark. In: SIGCOMM, Philadelphia, PA (August 2005)
3. Bernaille, L., Teixeira, R., Salamatian, K.: Early application identification. In: CoNEXT, Lisboa, PT (December 2006)
4. Zhang, M., Dusi, M., John, W., Chen, C.: Analysis of UDP Traffic Usage on Internet Backbone Links. In: Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet, Seattle, WA (July 2009)
5. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: ACM SIGCOMM, Pisa, IT (September 2006)
6. Erman, J., Mahanti, A., Arlitt, M.: Internet traffic identification using machine learning. In: IEEE GLOBECOM, San Francisco, CA (December 2006)
7. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow clustering using machine learning techniques. In: Barakat, C., Pratt, I. (eds.) PAM 2004. LNCS, vol. 3015, pp. 205–214. Springer, Heidelberg (2004)
8. Wang, Y., Xiang, Y., Yu, S.: An automatic application signature construction system for unknown traffic. Concurrency and Computation: Practice and Experience 22, 1927–1944 (2010)
9. Erman, E., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Semi-supervised network traffic classification. In: ACM SIGMETRICS, San Diego, CA (June 2007)
10. Yuan, J., Li, Z., Yuan, R.: Information entropy based clustering method for unsupervised internet traffic classification. In: IEEE ICC, Beijing, CN (May 2008)
11. Finamore, A., Mellia, M., Meo, M., Rossi, D.: KISS: Stochastic Packet Inspection Classifier for UDP Traffic. IEEE/ACM Transactions on Networking 18(5), 1505–1515 (2010)
12. Mantia, G.L., Rossi, D., Finamore, A., Mellia, M., Meo, M.: Stochastic Packet Inspection for TCP Traffic. In: IEEE International Conference on Communication - ICC, Cape Town, SA (May 2010)
13. Berkhin, P.: A survey of clustering data mining techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data, ch. 2, pp. 25–71. Springer, Heidelberg (2006)
14. Bianco, A., Mardente, G., Mellia, M., Munafò, M., Muscariello, L.: Web User-session Inference by Means of Clustering Techniques. IEEE/ACM Transactions on Networking 17(2), 405–416 (2009)
15. Finamore, A., Mellia, M., Meo, M., Munafò, M., Rossi, D.: Live Traffic Monitoring with Tstat: Capabilities and Experiences. In: Osipov, E., Kassler, A., Bohnert, T.M., Masip-Bruin, X. (eds.) WWIC 2010. LNCS, vol. 6074, pp. 290–301. Springer, Heidelberg (2010)
16. Ciullo, D., da Rocha Neta, A.G., Horvath, A., Leonardi, E., Mellia, M., Rossi, D., Telek, M., Veglia, P.: Network Awareness of P2P Live Streaming Applications: a Measurement Study. IEEE Transanctions on Multimedia 12(1), 54–63 (2010)