Uncovering Relations Between Traffic Classifiers and Anomaly Detectors via Graph Theory

Romain Fontugne, Pierre Borgnat, Patrice Abry, Kensuke Fukuda

Uncovering Relations Between Traffic Classifiers and Anomaly Detectors via Graph Theory

Romain Fontugne[1], Pierre Borgnat[2], Patrice Abry[2] and Kensuke Fukuda[3]

[1]The Graduate University for Advanced Studies, Tokyo, JP
[2]Physics Lab, CNRS, ENSL, Lyon, FR
[3]National Institute of Informatics / PRESTO JST, Tokyo, JP

Abstract. Network traffic classification and anomaly detection have received much attention in the last few years. However, due to the the lack of common ground truth, proposed methods are evaluated through diverse processes that are usually neither comparable nor reproducible. Our final goal is to provide a common dataset with associated ground truth resulting from the cross-validation of various algorithms. This paper deals with one of the substantial issues faced in achieving this ambitious goal: relating outputs from various algorithms. We propose a general methodology based on graph theory that relates outputs from diverse algorithms by taking into account all reported information. We validate our method by comparing results of two anomaly detectors which report traffic at different granularities. The proposed method succesfully identified similarities between the outputs of the two anomaly detectors although they report distinct features of the traffic.

1 Introduction

Maintaining network resources available and secured in the Internet is an unmet challenge. Hence, various network traffic classifiers and anomaly detectors (hereafter both called as classifiers) have been recently proposed. However, the evaluation of these classifiers usually lacks rigor, leading to hasty conclusions [1]. Since synthetic data is rather criticized and common labeled database (like the datasets from the DARPA Intrusion Detection Evaluation Program [2]) is not available for backbone traffic; researchers analyze real data and validate their methods by manual inspection, or by comparison with other methods. Our final goal is to provide a reference database by labeling the MAWI archive [3] which is a publicly available collection of real backbone traffic traces. Due to the difficulties faced in analyzing backbone traffic (e.g. lack of packet payload, asymmetric traffic), we plan to label the MAWI archive by cross-validating results from several methods based on different theoretical backgrounds. This systematic approach permits to maintain updated database in which recent traffic traces are regularly added, and labels are improved with upcoming algorithms. This database aims at helping researchers by providing a ground truth relative to the state of the art.
However, we face several complicated issues to reach our final goal. This article discusses the difficulties faced in relating outputs provided by distinct algorithms, and proposes a methodology to achieve it. This is an

important first step for labeling traffic data traces. The main contribution is to provide a general methodology to efficiently compare outputs exhibiting various granularities of the traffic. It uncovers the relations between several outputs by inspecting all information reported by classifiers and the original traffic. Also, the proposed method inherently groups similar events and permits to label quantity of traffic at once.

1.1 Related work

Usually ground truth is built by hand implying a lot of human work. Several applications have been proposed to assist humans and speed up this laborious task [4–6]. For example, GTVS [4] helps researchers by automating several tasks, and authors claim that a 30 minutes trace from a gigabyte link can be labeled within days. Since our purpose is to label a database containing 15 minutes traffic traces taken everyday for 9 years (MAWI archive) manual labeling is unpractical.

Alternatively, specialized network interface drivers have been recently proposed [1, 7] to label traffic while packets are collected. These drivers trace each packet and retrieve the corresponding application. Although these approaches are really promising to compute confident ground truth from Internet edges, it is not applicable for backbone traffic.

Closer to our work, Moore et al. [8] proposed an application combining nine algorithms that analyze different properties of the traffic. This application successfully achieved an accurate classification on a full payload packet traces recording both link directions. TIE is also an application designed to label traffic with several algorithms [9]. It computes *sessions* — i.e. flows, bi-directional flows, or traffic related to a host — from the original traffic and provides them to encapsulated classifiers. The final label for each *session* is decided from the labels provided by each classifier. Although these two applications are similar to our work, they do not solve the general problem of relating outputs from distinct algorithms. Indeed, both applications restrict classifiers to label only flows, ignoring all classifiers that operate at other granularities (e.g. packet, host...) and their benefits. Thus, they only deal with flows and bypass the problem addressed in this paper. Our work provides a more general approach that permits to combine results from any classifier. This issue has been only tackled in previous work; for example, Salgarelli et al. [10] also discuss the challenges faced by researchers in comparing performances of classifiers and proposed unified metrics to measure the quality of an algorithm. Although the need of common metrics in evaluating classifiers is crucial, we stress that these measures are not sufficient to compare classifiers outputs. For instance, let A and B be two distinct classifiers with the same true positive score (as defined in [10]: the percentage of flows that the classifiers labeled correctly) equal to 50% on a certain dataset. Let assume that the combination of A and B achieve 75% of true positive on the same dataset, then it will be interesting to know what kind of traffic A could identify that B could not (and vice versa).

2 Problem statement

Comparing outputs from several classifiers seems at first glance to be trivial, but in practice, it is a baffling problem. The main issue is that classifiers report different features of the traffic that are difficult to systematically compare. Hereafter, we define an *event* as any classifier decision to categorize a traffic (i.e. alarms from anomaly detectors or labels from traffic classifiers). Formally, an event e is a set of items $e = \{t_{begin}, t_{end}, f_1, ..., f_h\}$ where t_{begin}, t_{end} are timestamps respectively standing for the begin and the end of identified traffic, and other items f_i, $0 < i \leq h$ correspond to one of the following five traffic features: $\{srcIP, dstIP, srcPort, dstPort, protocol\}$. At least one traffic feature ($0 < h$) is required to describe identified traffic. For example, the event $e1 = \{t_{begin} : 90\text{s}, \text{t}_{\text{end}} : 150\text{s}, \text{srcPort} : 80\}$ refers to one minute of traffic from source port 80. Also, the same traffic feature can occur several times in a single event. For example the event $e_2 = \{t_{begin} : 30\text{s}, \text{t}_{\text{end}} : 90\text{s}, \text{srcPort} : 53, \text{protocol} : \text{udp}, \text{protocol} : \text{tcp}\}$ refers to one minute of UDP or TCP traffic from port 53.

2.1 Granularity of events

The traffic granularity of reported events results from the diverse traffic abstractions, dimensionality reductions and theoretical tools employed by classifiers. For example, in the case of anomaly detection:

- hash based (sketch) anomaly detectors [11, 12] usually report only IP addresses and corresponding time bin, no other information (e.g. port number) describes identified anomalies.
- An anomaly detector based on image processing reports an event as a set of IP addresses, port numbers and timestamps corresponding to a group of packets identified in analyzed pictures [13].
- Several intrusion detection systems take advantage of clustering techniques to identify anomalous traffic [14]. These methods classify flows in several groups and report clusters with abnormal properties. Thereby, events reported by these methods are sets of flows.

These different kinds of event provide distinct details of the traffic that are difficult to systematically compare. A simple way is to digest all of them to a less restrictive form; namely, by examining only the source or destination IP addresses (assuming that anomaly detectors report at least one IP address). Comparing only IP addresses permits to determine that *Event 1*, *Event 2* and *Event 3* in Fig. 1 are similar. However, the port numbers provided by *Event 2* and *Event 3* indicate that these two events represent distinct traffics. Consequently, an accurate comparison of these two events requires to also take into account port numbers, but it raises other issues. First, a heuristic is needed to make a decision when port number is not reported (for example in comparing *Event 1* and *Event 2*). Second, fuzzy equality is required to compare *Event 4* and *Event 5* of Fig.1. So forth, inspecting various traffic features reported by events makes the task harder although the accuracy of the comparison increases.

Similar problems arise in the case of traffic classification where different entities are labeled:

Fig. 1. Event 1, Event 2 and Event 3 report different traffics from the same host. A same port scan is reported by two events; Event 4 identifies only a part of it (beginning of the port range), whereas Event 5 identifies another part (the end of the port range).

- Usually flows are directly labeled (e.g. based on clustering techniques [15–17]).
- Whereas, BLINC [18] decides a label for a source (IP address, source port) based on its connection pattern. Also, a recent method [19] labels directly hosts without any traffic information by collecting and analyzing information freely available on the web.

Thus, researchers faced difficulties in comparing events standing for flows with events representing hosts. A common way is to apply the same label to all flows initiated from the host reported by an event, thus, only flows are compared [16]. Unfortunately, this manner to compare these two kinds of traffic classifiers leads to erroneous results. For example, if an host is reported by a classifier as a *web client* then all its corresponding flows are casted as *HTTP*. A simple port-based method also classifies most of these flows as *HTTP* but a few of them are labeled as *DNS*. In this case we cannot conclude that the port-based method misclassified DNS flows neither the other classifier failed in classifying this host. Obviously, the transition between an event representing host to its corresponding flows introduce errors. More sophisticated mechanisms are required to handle this two concepts (flow and host), whereas the synergy between them might provides accurate traffic classification.

2.2 Traffic assortment

Recent applications and network attacks tend to be distributed over the network and composed of numerous flows. Therefore, classifiers labeling flows output an excessive number of events for traffic generated by distributed applications. Regardless the quantity of traffic involved by a unique attack, or instance of an application, the whole amount of generated traffic should be handled and annotated as a single entity. Thus, traffic annotations are clarified and highlight connection pattern of hosts. In some cases, finding these similarities between events requires to retrieve the original traffic. For example, let X be an event corresponding to traffic emitted from a single host, and Y an event representing traffic received by another host. X and Y can represent exactly the same traffic but from two different points of view, one reports the source whereas the other one reports the destination of the traffic. The only way to verify if these events are related to each other is to investigate the analyzed traffic. If all traffic reported by X is also reported by Y, then we conclude that they are strongly related. Obviously, a quantitative measure is also needed to accurately score their similarities.

3 The proposed method

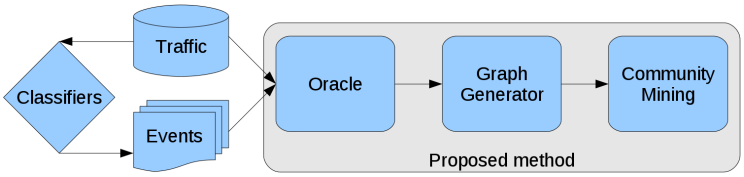

Fig. 2. Overview of the proposed method.

We propose a method to relate several events of different granularities by analyzing all their details. The main idea underlying our approach is to discover events relations among events from original traffic (oracle in Fig.2) and represent all events and their relations as a graph (graph generator in Fig.2). Afterwards, coherent groups of similar events are identified in the graph with an algorithm finding community structure (community mining in Fig.2).

3.1 Oracle

The oracle is the interface binding specific classifiers outputs to our general methodology. Its role is to retrieve the relation between the original traffic and the reported events. It accepts a query in the form of a packet p and returns a list of events, $R_p = \{e_{p0}, ..., e_{pn}\}$, consisting of all events from every classifiers that are relevant to the query. Formally, let a packet p be a set of five distinct traffic features and a timestamp t_p, $p = \{t_p, f_{p1}, ..., f_{p5}\}$ then R_p consists of all events $e = \{t_{begin}, t_{end}, f_1, ..., f_h\}$ where $t_{begin} \leq t_p \leq t_{end}$ and $\exists f_j, f_j = f_{pi}$, with $0 < i \leq 5$ and $0 < j \leq h$. Queries are generated for each packet of the original traffic, thereby the oracle produces the lists of events matching all packets $R = \{R_{p1}, ..., R_{pm}\}$ (m is the total number of packets).

3.2 Graph Generator

The graph generator collects all responses from the oracle and build a graph highlighting event similarities. Nodes of the graph represent the events and those appearing in a same list returned by the oracle are connected to each other by edges. Thus, for any edge of the graph (e_x, e_y) there is at least one list provided by the oracle, $R_{pz} \in R$, in which the two connected edges appear $e_x, e_y \in R_{pz}$. Weights of edges quantify the similarities of events based on the quantity of traffic they have in common. Let $c(e_1, ..., e_n)$ be a function returning the number of lists, $R_{pz} \in R$, in which all events given as parameters appear together, $e_1, ..., e_n \in R_{pz}$. Then the weight of an edge (e_x, e_y) is computed with the following equation:

$$w(e_x, e_y) = c(e_x, e_y) / \min(c(e_x), c(e_y))$$

w ranges $(0, 1]$, 1 means that events are strongly related whereas values close to 0 represent weak relationships.
The characteristic of graphs built by the graph generator is that connected components stand for sets of events representing common traffic. Also, connected components consists of sparse and dense parts, hereafter, we define a community as a coherent group of nodes representing similar events.

3.3 Community mining

The next step is to find out community structure [20] to identify coherent groups of similar events within connected components of graphs constructed by the graph generator. Although many kinds of community structure algorithm have been proposed, we only take an interest in those based on modularity because there exists versions that perform fast on sparse graph [21].

Modularity Newman and Girvan proposed a metric for evaluating the strength of a given community structure [20] based on inter and intra-community connections; this metric is called the *modularity*.
The main idea underlying the modularity is that the fraction (or proportion) of edges connecting nodes of a single community is expected to be higher than the value of the same quantity in a similar graph where nodes are randomly connected. Let e_{ij} be a fraction of edges connecting nodes of community i to those of community j, such that e_{ii} is the fraction of edges within a community i. Thus, $\sum_i e_{ii}$ is the total fraction of edges connecting nodes of the same community. This value highlights the connections within communities, a large value represents a good division of the graph in communities. However, it takes the maximum value 1, for particularly meaningless case in which all nodes are grouped in a single community.
Newman et al. enhanced this measure by subtracting from it the value it would take if edges were randomly placed. We note $a_i = \sum_j e_{ij}$ the fraction of all edges attached to nodes in community i. If the edges are placed at random, the fraction of edges that link nodes within community i is a_i^2. The modularity is defined as $Q = \sum_i (e_{ii} - a_i^2)$.
If the fractions of edges within communities are similar to those expected in a randomized graph, then score of the modularity will be 0, whereas $Q = 1$ indicates graphs with strong community structure. Since Q represents the quality of community structure in a graph, researchers investigated this metric to efficiently partition graph in communities.

Finding communities Blondel et al. proposed an algorithm [21] finding community structure by optimizing the modularity in an agglomerative manner.
Their algorithm starts by assigning a community to each node of the graph, then the following step is repeated iteratively. For each community i the gain of modularity obtained by merging it with one of its neighbor

j is evaluated. The merge of i and j is done for the maximum gain, but only if this gain is positive. Otherwise i is not merged with other communities. Once all nodes have been examined a new graph is build, and this process is repeated again until no merge can be done.
The authors claim that the computational complexity of their algorithm is linear on typical and sparse data. In their experiments Blondel et al. successfully analyzed a graph with 118 million nodes and 1 billion edges in 152 minutes. The performances of this algorithm allow us to compare thousands of events in a really short time frame (order of seconds).

4 Evaluation

4.1 Data and processing

The proposed method is preliminarily evaluated by comparing the results of two anomaly detectors based on different theoretical backgrounds.
One consists of random projection techniques (sketches) and multi-resolution gamma modeling [11]. Hereafter we call it as the gamma-based method. In a nutshell, the traffic is split into sketches and modeled using Gamma laws at several time scales. Anomalous traffic is detected by reporting too large distances from an adaptively computed reference. The sketches are computed twice; the traffic is hashed on source addresses and on destination addresses. Thus, when anomalies are detected this method reports the corresponding source or destination address within a certain time bin.
The other anomaly detector is based on an image processing technique called the Hough transform [13] (we call it the Hough-based method). Traffic is monitored in 2-D scatter plot where each plot represents packets and anomalous traffics appear as "lines". Anomalies are extracted with a line detector (the Hough transform) and the original data are retrieved from the identified plots. The output of this method is an aggregated set of packets.
These two anomaly detectors were tested on a pcap file of the MAWI archive containing 15 minutes of traffic taken at a trans-Pacific link between Japan and US (Samplepoint-B, 2004/08/01) corresponding to a period of Sasser outbreak.
In practice, the output of these two anomaly detectors is in admd[1] form, which is a XML schema allowing to annotate traffic in an easy and flexible way. Hence, we implemented an oracle able to read any admd and pcap file to compare results from both methods.

4.2 Results

In our experiments 332 events have been reported by the gamma-based method and 873 by the Hough-based one, where respectively 235 and 247 events have been merged by our method. The resulting graph consists of 124 connected components (we do not consider components with a single event), we present some typical graph structures in this Section.

[1] Meta-data format and associated tools for the analysis of pcap data: http://admd.sourceforge.net

(a) Both methods detect the same host infected by the Sasser worm.

(b) The gamma-based method reports the destination of anomalous traffic whereas the Hough-based one reports the source of it.

Fig. 3. Two simple connected components with two similar events.

Note that we use following legend for Fig. 3-7. Gray rectangles represent the separation in community structure, green ellipses are events reported by the Hough-based method, and red rectangles are events reported by the gamma-based method. The labels of events are displayed as: *IPaddress direction;nbPackets*, where *IPaddress* is the IP address reported by the gamma-based model or the prominent IP address of traffic reported by the Hough-based method; *direction* is a letter, *s* or *d*, informing if the identified hosts are rather the sources or destinations of reported traffic; *nbPackets* is the number of packets in the traffic trace that match the event. We emphasize that the IP addresses shown in these labels are only provided to facilitate the readability of figures. Thus it is not the only information considered in the oracle decisions (the gamma-based method also reports timestamps, and the Hough-based method can provide several IP addresses, port number, timestamps, and protocols). The label for an edge linking two events is the weight of the edge w and the number of packets matching both events.

Simple connected components Figure 3 consists of two examples of the simplest connected components built by our method. Figure 3(a) stands for the same Sasser activity reported by both methods. Since the two methods reported anomalous traffic from the same host, the events have been obviously grouped together. The single edge between the two events represents numerous incomplete flows that can be labeled as the same malicious activity. Figure 3(b) displays two events reporting different hosts; one event describes anomalous traffic sent from a source IP (172.92.103.79), whereas the other one exhibits abnormal traffic received by another host (210.133.66.52). Their relationship is uncovered by the original traffic, all packets (except one) initiated from the source host have been sent to the identified host destination. This connected component illustrates the case where both anomaly detectors report the same

traffic but from different points of view, one identified its source whereas the other emphasized its destination.
In our experiments, we observed 86 connected components containing only 2 events (like those depicted in Fig.3) where, the two linked events are sometimes reported by the same anomaly detector.

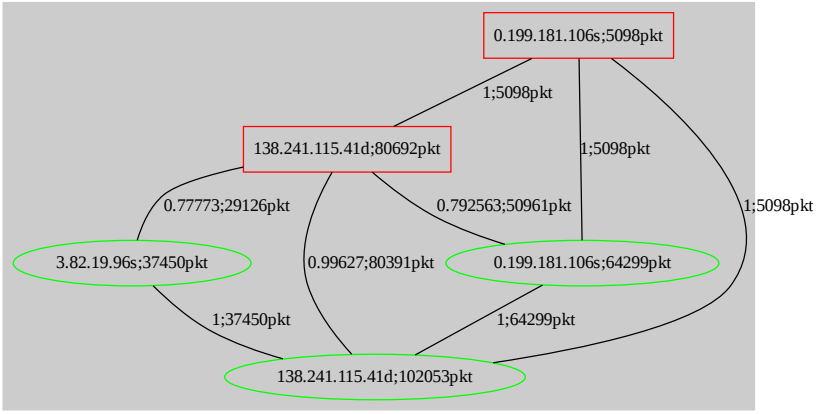

Fig. 4. RSync traffic identified by 5 events.

Large connected components The proposed method found 38 connected components that consist in more than 2 events. For example, Fig. 4 shows 5 events grouped in a strongly connected component. All these events report abnormally high traffic volume, and manual inspection of packets header revealed that they are all RSync traffic. Three hosts are concerned by these events, and the structure of the component help us in understanding their connection pattern. The weights of edges indicate that these events are closely related. Thus, these 5 events are grouped as one community that is uniquely reported.
Figure 5 depicts a connected component consisting of 29 events; 27 are from the gamma-based method output and 2 from the Hough-based one. All these events are reporting abnormal DNS traffic. The event on the right-hand side of Fig.5 and the one on the left-hand side (both labeled 200.24.119.113) represent traffic from a DNS server. This server is reported by both methods because it replies to numerous requests during the whole traffic trace. Other events shown in Fig.5 represent the main clients soliciting this service. By grouping all these events together our method permits to report the flooded server and the blamed clients at the same time. Whereas, by analyzing individually events raised by clients,

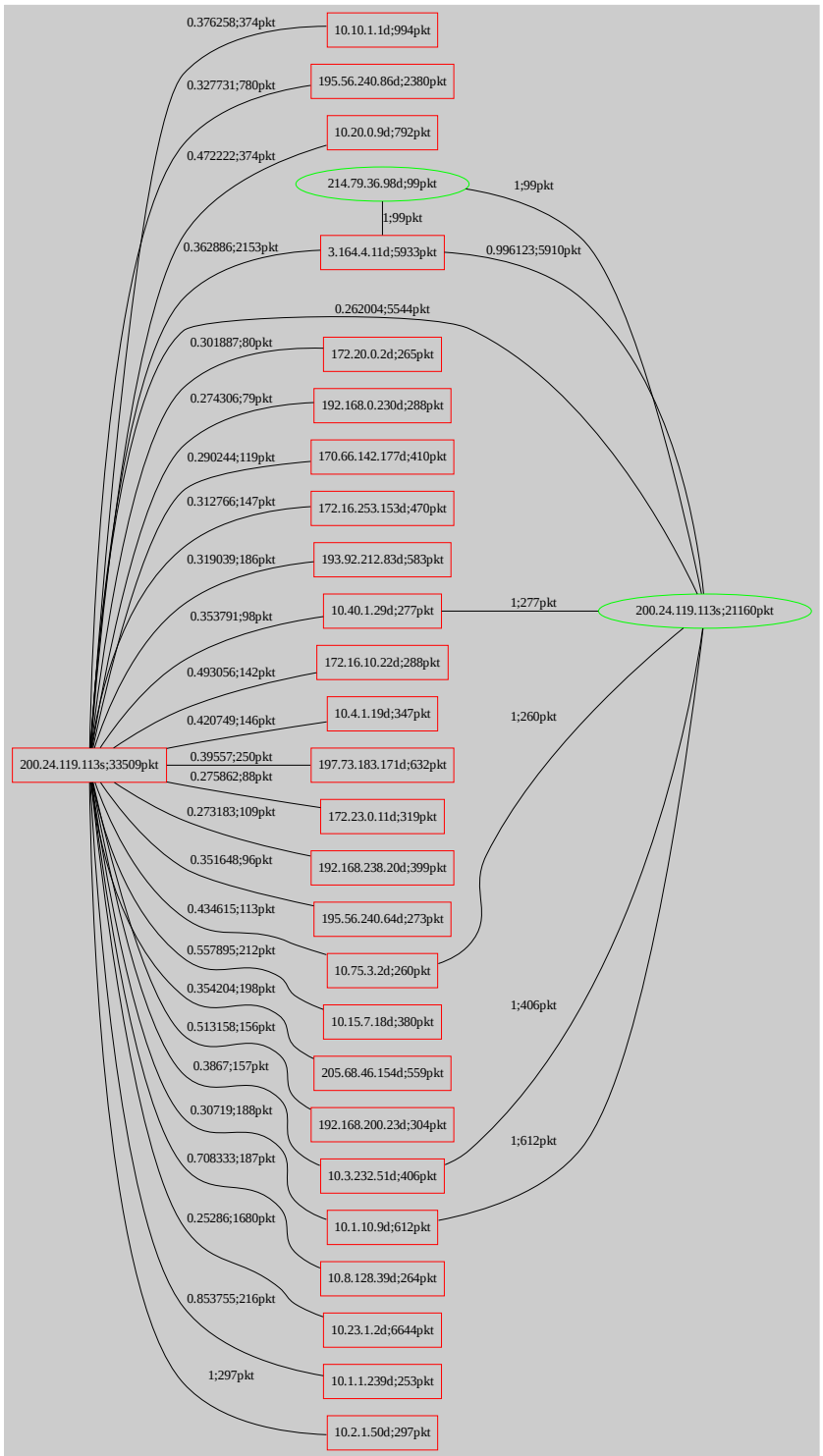

Fig. 5. DNS traffic reported by many events.

one may misunderstand the distributed characteristic of this network activity — similar to DDoS, flash crowd, or botnet activity — and misinterpret each event.

Communities in connected components In the examples presented above the algorithm finding community structure (see Section 3.3) identified each connected component as a single community. Nevertheless, our method found 11 connected components that are split in several communities (e.g. Fig. 6 and 7); the smallest contains 5 events grouped in 2 communities, and the largest consists of 47 events clustered in 8 communities. These connected components stand for distinct network traffics that are linked by loose events (i.e. events reporting only one traffic feature). Fortunately, the algorithm finding community structure succeed in cutting connected components in coherent groups of events.
An example of a connected component representing two communities is depicted in Fig.6. The community on the left-hand side of Fig.6 stands for a high-volume-traffic directed to port number 3128 (proxy server). However, the community on the right-hand side of Fig.6 represents nntp traffic between two hosts. A single packet is responsible for connecting two events from both communities. It is a TCP/SYN packet sent from the main host representing the left-hand side community and aiming at the port 3128 of a host belonging to the other community. This is the only traffic observed on port 3128 for the latter host. The proposed method successfully dissociates the two sets of events having no similarities, so they can be handle separately.
Figure 7 depicts another example of connected component split in several communities, but this involves 14 events grouped in 5 communities. All events report uncommon HTTP traffic among numerous hosts. Although all events are connected together, weight of edges emphasizes several dense sets of events. By analyzing the weight of edges and the degree of nodes, the algorithm finding community structure successfully detected these coherent groups of events.

4.3 Discussion

The proposed method enabled us to compare outputs from different kinds of classifier (e.g. host classification and flow classification), and fulfill our requirements to combine results form many classifiers.
Our method is also useful in inspecting the output of a single method. For example, the gamma-based method inherently reports either source or destination address of anomalous traffic, but both are sometimes reported in two distincts events. Let T be an anomalous traffic between hosts A and B raising two events $e_x = \{t_{begin} : X, t_{end} : Y, srcIP : A\}$ and $e_y = \{t_{begin} : X, t_{end} : Y, dstIP : B\}$, then the proposed method merges these events as packets $p \in T$ are in the form $p = \{t_p : Z, srcIP : A, dstIP : B, ...\}$ with $X \leq Z \leq Y$. In our experiments, our method permits to merge 27 events (see Fig.5) reported by the gamma-based method increasing the quality of reported events and reducing the size of the output.

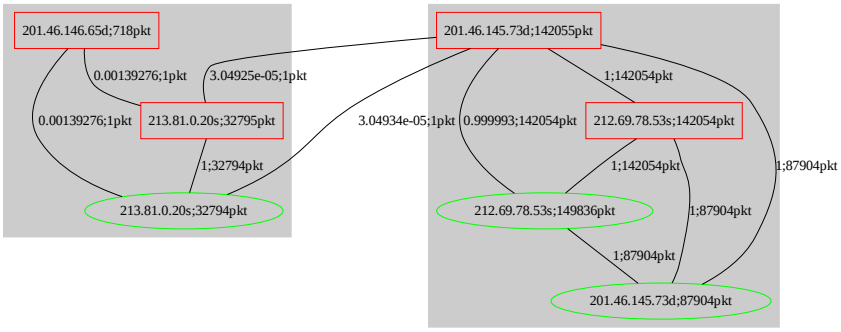

Fig. 6. Connected component standing for two distinct traffics.

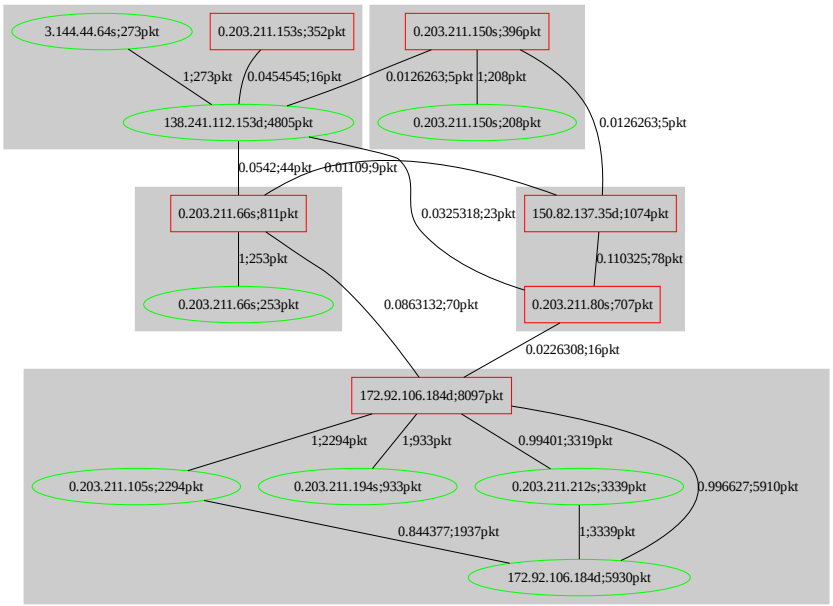

Fig. 7. HTTP traffic represented by a large connected component split in 5 communities.

Another benefit of the proposed method is to help researchers in understanding different results from their algorithms. For instance, while developing anomaly detector, researchers commonly face a problem in tuning their parameter set. Therefore, researchers usually run their application with numerous parameter settings, and the best parameter set is selected by looking at the highest detection rate. Although this process is commonly accepted by the community a crucial issue still remains. For instance, a parameter set A may give a similar detection rate to that obtained with a parameter set B, but a deeper analysis of reported events may show that B is more effective for a certain kind of anomalies not detectable with the parameter set A (and vice versa). Deciding if A or B is the best parameter is then not straightforward. This interesting case is not solved by simply comparing detection rates. The overlap of both outputs as exhibited by our method would help us first to compare in which conditions a parameter set is more effective, second to make methods collaborate.

5 Conclusion

This article first raised the difficulties in relating outputs of different classifiers. We proposed a methodology to relate reported events although they are expressed in different ways and represent distinct granularities of the traffic. Our approach relies on the abstraction level of graph theory, graphs are generated from events and the original traffic to uncover the similarities of events. An algorithm finding community structure permits to distinguish coherent sets of nodes in the graph standing for sets of similar events. Preliminary evaluation highlighted the flexibility of our method and its effectiveness to cluster events reported by different anomaly detectors.
The proposed methodology is a first step in our process to build a common database of annotated backbone traffic. We need more analyses to better understand the basic ability of the proposed method with different datasets and classifiers. In future work we will also adopt a strategy taking into account the nature of classifiers to decide the final label to annotate traffic represented by a set of events.

Acknowledgments

We would like to thank V.D. Blondel et al. for having provided us with the source code of their community structure finding algorithm. This work is partially supported by MIC SCOPE.

References

1. Szabó, G., Orincsay, D., Malomsoky, S., Szabó, I.: On the validation of traffic classification algorithms. PAM '08 (2008) 72–81
2. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 darpa off-line intrusion detection evaluation. Computer Networks **34**(4) (2000) 579 – 595

3. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository at the WIDE project. In: USENIX 2000 Annual Technical Conference: FREENIX Track. (June 2000) 263–270
4. Canini, M., Li, W., Moore, A.W., Bolla, R.: Gtvs: Boosting the collection of application traffic ground truth. TMA '09 (May 2009)
5. Haakon Ringberg, A.S., Rexford, J.: Webclass: adding rigor to manual labeling of traffic anomalies. SIGCOMM CCR **38**(1) (2008) 35–38
6. Fontugne, R., Hirotsu, T., Fukuda, K.: A visualization tool for exploring multi-scale network traffic anomalies. SPECTS '09 (2009) 274 – 281
7. Gringoli, F., Salgarelli, L., Cascarano, N., Risso, F., Claffy, K.C.: Gt: Picking up the truth from the ground for internet traffic. SIGCOMM CCR **39**(5) (2009) 13–18
8. Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. PAM '05 (2005) 41–54
9. Dainotti, A., Donato, W., Pescapé, A.: Tie: A community-oriented traffic classification platform. TMA '09 (May 2009) 64–74
10. Salgarelli, L., Gringoli, F., Karagiannis, T.: Comparing traffic classifiers. SIGCOMM CCR **37**(3) (2007) 65–68
11. Dewaele, G., Fukuda, K., Borgnat, P., Abry, P., Cho, K.: Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. SIGCOMM LSAD '07 (2007) 145–152
12. Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., Lakhina, A.: Detection and identification of network anomalies using sketch subspaces. SIGCOMM '06 (2006) 147–152
13. Fontugne, R., Himura, Y., Fukuda, K.: Evaluation of anomaly detection method based on pattern recognition. IEICE Trans. on Commun. **E93-B**(2) (February 2010)
14. Sadoddin, R., Ghorbani, A.A.: A comparative study of unsupervised machine learning and data mining techniques for intrusion detection. MLDM '07 (2007) 404–418
15. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. SIGCOMM MineNet '06 (2006) 281–286
16. chul Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: Internet traffic classification demystified: Myths, caveats, and the best practices. CoNEXT '08 (2008)
17. Bernaille, L., Teixeira, R., Salamatian, K.: Early application identification. CoNEXT '06 (2006) 1–12
18. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: Blinc: multilevel traffic classification in the dark. SIGCOMM '05 **35**(4) (2005)
19. Trestian, I., Ranjan, S., Kuzmanovi, A., Nucci, A.: Unconstrained endpoint profiling (googling the internet). SIGCOMM '08 (2008)
20. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2) (Feb 2004) 026113
21. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J.STAT.MECH. (2008)