Infrastructure description - Related work, basic architecture and algorithms design for online traffic characterization (ONTIC D4.1)

Philippe Owezarski, Bruno ; Ordozgoiti, Alberto Mozo, Alexandru Mara

Online Network Traffic Characterization

Deliverable
Infrastructure description
Related work, basic architecture and algorithms design for online traffic characterization

ONTIC Project
(GA number 619633)

Deliverable D4.1
Dissemination Level: PUBLIC

Authors

Owezarski, Philippe
CNRS

Ordozgoiti, Bruno
Mozo, Alberto
Mara Alexandru
UPM

Version

ONTIC_D4.1.2015.01.29.1.0

Version History

Previous version	Modification date	Modified by	Summary
0.1	2014.12.24	CNRS	1st draft version
0.2	2015.01.15	UPM	Included contributions from UPM
0.3	2015.01.16	UPM	Format adjusted
0.4	2015.01.20	CNRS	2nd draft version
0.5	2015.01.28	CNRS	1st reviewed version
0.6	2015.01.29	CNRS	2nd reviewed version
1.0	2015.01.29	SATEC	Final version

Quality Assurance:

	Name
Quality Assurance Manager	**Miguel Angel López (SATEC)**
Reviewer #1	**Sotira Chatzi (ADAPTIT)**
Reviewer #2	**Daniele Apiletti (POLITO)**

Table of Contents

List of figures

Acronyms and Definitions

Acronyms

Acronym	Defined as
ARMA	AutoRegressive Moving Average
ARIMA	AutoRegressive Integrated Moving Average
BLRNN	Bilinear Recurrent Neural Network
DBSCAN	Density Based SCANning
DoS	Denial Of Service
DDoS	Distributed DoS
CF	Clustering Features
CFT	Clustering Features with a Temporal component
DWT	Discrete Wavelet Transform
EA	Evidence Accumulation
EAC	Evidence Accumulation Clustering
EHCF	Exponential Histogram of Cluster Features
FES	Feature Engineering System
FIR	Finite Impulse Response
FIRNN	Finite Impulse Response Neural Network
FS	Fisher Score
ICMP	Internet Control Message Protocol
IDS	Intrusion detection System
IP	Internet Protocol
IPdst	Destination IP address
IPsrc	Source IP address
ISP	Internet Service Provider
KDD	Knowledge Discovery and Data mining
nDsts	Number of destinations
nICMP	Number of ICMP packets
nPkts	Number of packets
nSrcs	Number of sources

nSYN	Number of SYN packets
OD	Origin-Destination
PCA	Principal Components Analysis
RLS	Recursive Least Square
SA	Streaming Algorithm
SPS	Stream Processing System
SSC	Sub-Space Clustering
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
SYN	Synchronization TCP packet
TCP	Transmission Control Protocol
TpTS	Traffic per Time Slot
UNADS	Unsupervised Network Anomaly Detection System
URCA	Unsupervised Root Cause Analysis
WP	Work Package

1. Purpose of the Document

Deliverable D4.1 describes the preliminary work performed on the design of the online traffic characterization system. It first draws the state of the art of the existing work related with online traffic characterization and anomaly detection, especially focusing on the full range of approaches going from supervised to unsupervised ones. The deliverable also defines the basics of the generic online characterization system and the specificities of the two main topics addressed in WP 4 namely traffic pattern evolution analysis and anomaly detection. It particularly indicates how this architecture takes advantage of the big data infrastructure as developed in WP2. Deliverable 4.1 also indicates the principles of the algorithms for both traffic pattern evolution analysis and anomaly detection. Moreover, it creates the link with use case 1 (Network Anomaly detection) of WP5, indicating how the algorithm evaluation could be performed.

2. Scope

This document provides the description of the current status of work lead in WP4 on the online traffic classification system. The progress in ONTIC on the online traffic classification system, traffic pattern evolution analysis, and anomaly detection during the second year will be held in Deliverable 4.2, as a continuation of the current deliverable. The third and last year will generate deliverable 4.3 aiming at presenting the results of the intensive evaluation of the algorithms designed for the online classification system and related applications, i.e. pattern evolution subsystem, and anomaly detection.

3. Intended Audience

The intended audience includes every partner within ONTIC project, especially those involved in WP2, WP3, WP4 and WP5.

4. Suggested Previous Readings

There are some overview papers that might be interesting to get started in machine learning:

- Tom M. Mitchell, “The Discipline of Machine Learning”. This is a white paper defining the discipline of Machine Learning. This was a piece of the argument Mitchell used to convince the President of CMU to create a standalone Machine Learning department for a subject that will still be around in 100 years. The paper can be found at the following URL: http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf
- Pedro Domingos, “A Few Useful Things to Know about Machine Learning”. This is a great paper because it pulls back from specific algorithms and motivates a number of important issues such as feature selection generalizability and model simplicity. This is all good stuff to get right and think clearly about from the beginning. The paper can be found at the following URL: http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

The following book is also good for getting started to basic Machine Learning algorithms applied to Data Mining:

- Ian H. Witten, Eibe Franck, Mark A. Hall, “Data Mining: Practical Machine Learning Tools and Techniques, Third Edition”, The Morgan Kaufmann Series in Data Management System, ISBN-13: 978-0123748560 ISBN-10: 0123748569, January 2011

Last, it might be interesting to follow the online lecture called “Stanford Machine Learning”. It is Available via Coursera and taught by Andrew Ng. The course includes homework and quizzes and focuses on linear algebra and using Octave.

5. Executive Summary

ONTIC aims at proposing a framework composed of techniques that can mostly be qualified as unsupervised (or semi supervised in some cases) for traffic characterization with direct applications that are:

a) Able to catch traffic patterns evolution
b) Able to detect network anomalies and especially attacks.

This system has to run online with a reactive level close to real-time. It then has to cope with the huge amount of data and their high dimensionality level. In addition, this system consists of several functions (detection and analysis of traffic patterns evolution, anomalies and intrusions detection, automatic defense devices configuration with autonomously generated filtering rules, continuous updating of the system knowledge database, visualization, etc.) that have to run continuously and in parallel: parallelizing and synchronizing the execution of these functions is therefore an essential feature of the system. As the subsystems have to mainly rely on unsupervised techniques (such as clustering), we have investigated a scalable and elastic architecture for online data stream clustering for which more/less processes can be seamlessly added or removed on the fly during computation. A specific architecture for all these functionalities is being designed in order to reach the near real-time reaction objective. It is based on sub-space clustering that allows parallelization of the computing in all subspaces, as they are completely independent. It is then possible with powerful enough parallel machines to theoretically benefit from a maximum speedup. Subspace clustering also limits the dramatic impact on the clustering results of the noise inherent to large and high dimensionality spaces. The noise is being the source of inaccuracy and inefficiency in many existing clustering techniques that limit their capabilities for analyzing large sets of data as traffic flowing in high-speed networks. This point is then at the heart of researches in the domain of clustering algorithms for some time, with more or less valuable results that often depend on kind of data to me mined. The subspace clustering technique is being designed with the objective of overpassing other ones in terms of accuracy, efficiency and speed in all situations as it is natively designed to cope with the huge amount of data flowing in high-speed networks and their related high dimensionality level.

On the other side, when traffic patterns evolution is detected, a supervised subsystem can be of interest working in conjunction with the unsupervised one. After classes of traffic, including possible new ones have been extracted from the traffic, the supervised system can be re-trained for updating the system knowledge database. It could also help the intrusion detection subsystem for improving its efficiency by reducing the analysis domain in each case.

Last, the anomaly detection subsystem aims at autonomously detecting anomalies (including the ones due to attacks), and to autonomously trigger countermeasures. For this purpose we are investigating data mining techniques for autonomously generating anomalies characteristics. In the following months of the ONTIC project we will investigate information theory techniques for giving a score to anomalies abnormality, and depending on the score, generate automatically filtering rules to be deployed on network security devices. The target is to have very fast reaction giving the feeling to administrators and users of a proactive defense system.

6. Related work

6.1 From supervised to unsupervised traffic classification and anomaly detection

The problem of network anomaly detection has been extensively studied during the last decade. Most of the approaches analyze statistical variations of traffic volume (e.g. number of packets, bytes or new flows) and/or traffic features (e.g. IP addresses and ports), using either single-link measurements or network-wide data. A non-exhaustive list of standard methods includes the use of signal processing techniques (e.g. ARIMA - Autoregressive Integrated Moving Average - modeling, wavelets-based filtering) on single-link traffic measurements [1][2], PCA (Principal Component Analysis) for network-wide anomaly detection [4][5][6], and Sketches applied to IP-flows [3][7].

The simultaneous detection and characterization of traffic anomalies has also received quite a lot of attention in the past, but results are few and present important limitations, either because they rely on some kind of training data and/or anomaly signatures, or because they do not provide meaningful and tractable information to a human network operator, who has to take the final decision about the nature of the detected problem. Authors in [4] characterize network-wide anomalies in highly aggregated traffic (Origin-Destination flows or OD flows for short), using PCA and the sub-space approach [6]. An important limitation of this approach is that the information obtained from OD flow data is too coarse-grained to provide meaningful information to the network operator. Papers like Lakhina et al. [5] and Biang et al. [7] detect and characterize anomalies using finer-grained traffic information, basically applying the same PCA approach to the sample entropy of the empirical distribution of specific traffic features. One clear limitation of these approaches is that the information they provide is not immediately usable and easy-to-understand by the network operator, who may not even be familiar with concepts distant from his tasks such as sample entropy. Besides, the PCA approach is highly sensitive to noise when used for anomaly detection [8][9], requiring in practice a fine-tuning and data-dependent calibration step to work.

UNADA (Unsupervised Network Anomaly detection Algorithm) [10] falls within the unsupervised anomaly detection domain, a novel research area that has drawn quite a lot of interest in the research community, but that still represents a rather immature field. Most work on unsupervised network anomaly detection has been devoted to the IDS field, generally targeting the detection of network intrusions in the very well-known KDD'99 dataset. The great majority of the detection schemes proposed in the literature are based on clustering techniques and outliers detection, being [11][12][13] some examples. The objective of clustering is to partition a set of unlabeled patterns into homogeneous groups of “similar'” characteristics, based on some similarity measure. Outliers detection consists in identifying those patterns that do not belong to any of these clusters. In [13], authors use a simple single-linkage hierarchical clustering method to cluster data from the KDD'99 dataset, based on the standard Euclidean distance for inter-pattern similarity. Eskin et al. [11] reports improved results in the same dataset, using three different clustering algorithms: the Fixed-Width clustering algorithm, an optimized version of the K-NN algorithm, and the one-class SVM algorithm. Leung and Leckie [12] present a combined density-based and grid-based clustering algorithm to improve computational complexity, obtaining similar detection results.

Previous work of some ONTIC partners permits to automatically characterize network traffic anomalies [14], but using a-priori well-defined anomaly signatures. Closer to our current work, authors in [15] present URCA (Unsupervised Root Cause Analysis), a two-steps

algorithm to characterize network anomalies in an unsupervised fashion. URCA uses as input the traffic in the anomalous time slots detected by any generic time-slot-based detection algorithm [16]. In the first step, it identifies the anomaly by iteratively removing from the anomalous time slots those flows that seem normal. In the second step, the algorithm uses a hierarchical clustering method to characterize the particular flows identified as anomalous. We identify some serious drawbacks and omissions in URCA: authors claim that the approach is unsupervised, which is not true, as it actually uses previously labeled anomalous events for the characterization. As in previous works, the algorithm uses difficult-to-interpret traffic descriptors for the clustering step (e.g. sample entropy of the distribution of IP addresses, aggregated at different levels), obscuring the comprehension of the network operator. Finally, the algorithm removes those flows that seem normal before the characterization step, which drags possible errors to the clustering step.

The Unsupervised Anomaly Detection and Characterization algorithm [17] from some ONTIC partners presents several advantages with respect to current state of the art. First and most important, it works in a completely unsupervised fashion, which means that it can be directly plugged into any monitoring system and start to work from scratch. Secondly, anomaly detection is performed based not only on outliers detection, but also by identifying small-clusters. This is achieved by using different levels of traffic aggregation, both at the source and destination of the traffic; this additionally permits to discover low-intensity and distributed anomalies. Thirdly, the lack of robustness of general clustering approaches is avoided by combining the notions of Sub-Space Clustering [18] and multiple Evidence Accumulation [19]. In particular, this algorithm is immune to general clustering problems like sensitivity to initialization, specification of number of clusters, or structure-masking by irrelevant features. Fourthly, the algorithm performs clustering in low-dimensional feature spaces, using simple traffic descriptors like number of source IP addresses or fraction of SYN packets. This simplifies the characterization of the anomaly, and avoids well-known clustering problems when working with high-dimensional data [20]. This algorithm ranks the multiple evidence of an anomaly detected in different sub-spaces, combining the most relevant traffic descriptors into a compact and easy-to-interpret signature that characterizes the problem. This permits to reduce the time spent by the network operator to understand the nature of the anomaly. Finally, this algorithm is designed to work in an on-line fashion, analyzing traffic from consecutive time slots in near real time. This is possible even when working with large number of traffic descriptors, because the sub-space clustering and the evidence accumulation algorithms are perfectly adapted for parallelization (see [17]).

6.2 Traffic pattern evolution

As stated in the previous section, unsupervised traffic analysis has received a lot of attention during the past decade. In this section, some of the main approaches and recent proposals for the application of these techniques to the problem of pattern evolution detection, analysis and prediction are discussed.

6.2.1 Online Clustering

Clustering is one of the most popular approaches for unsupervised learning. There exist different types of clustering algorithms:

- Partitioning algorithms, which divide a set of objects into different clusters with the goal of minimizing an objective function. An example is the classic k-means.

- Micro-clustering: they consist of two phases, a preliminary online micro-clustering of the data and a grouping into global clusters. Examples are BIRCH and CluStream.
- Density-based algorithms based on connectivity and density functions. Examples are DBSCAN and OPTICS.
- Grid-based algorithms, based on a multiple-level granularity structure, such as Fractal Clustering and STING, CLIQUE and MAFIA.
- Model-based algorithms, which aim to fit a model as well as possible, such as COBWEB and Kohonen's Self-Organized Maps.

Most classic clustering schemes generally rely on multiple passes over the data to reach convergence. Moreover, they group the data in hyperspheres, and the number of clusters requires to be set a priori by the user. These characteristics have very important drawbacks: (1) Performing multiple passes over big data sets is impractical, and often infeasible, (2) the constantly streaming, pseudo-infinite nature of many data sources nowadays completely rules out the possibility of multiple passes, (3) Gaussianity is necessary for hyperspheric clusters to be accurate and (4) prior knowledge about the expected number of clusters is necessary for the algorithm to perform well. In addition, clustering algorithms in an online setting must address the problems of concept drift, concept shift and ageing. There have been numerous attempts to overcome these issues for the applicability of clustering methods to online data streams, which undoubtedly constitutes an active research topic today. Some of the most relevant proposals in this field are presented below.

6.2.1.1 Micro Clustering

In [38], the concept of Micro-Clustering is described, along with the CluStream algorithm. This algorithm builds upon notions previously proposed by Single-Pass k-Means [42] and BIRCH [65]. The system is divided into two components, one online and one offline. The online phase constructs Clustering Features (CF), which represent a small set of data points, taking temporal information into account (CFT). Incoming points are absorbed by a CF if they fall within its maximum boundary. Otherwise, a new CFT is created. A maximum level of clusters is kept, removing old ones in favor of newer ones. CFTs can be stored periodically and aggregated in pyramidal form for offline clustering with different levels of granularity. The notion of Micro-Clustering (i.e. summaries to describe the data more compactly) has been used and developed extensively in posterior streaming clustering techniques.

6.2.1.2 Partitioning Clustering

Several researchers have proposed variations on the classic k-Means formula to enable the clustering of data streams. Two of the first notable proposals in this area are Scalable k-Means [41] and Single-pass k-Means [42]. In [43], O'Callaghan et al. propose STREAM, that analyzes arriving chunks (i.e. subsets of the stream that can fit into memory) in search of distinct points, which are then clustered using an algorithm called LOCALSEARCH. More recent works include StreamKM++ [44]. In this paper, the authors present a streaming version of the k-Means++ algorithm and introduce the concept of coreset tree. Coresets are small sets that yield similar clustering results to those of much larger sets, i.e. they are compact representations that retain features relevant for clustering. According to the experimental results presented, the algorithm is slower than BIRCH [65] and Stream [45] but yields better results in terms of squared errors. In [46] an algorithm for tracking cluster changes over sliding windows, SWCluster, is described. They propose a new data structure, the Exponential Histogram of Cluster Features (EHCF), to keep track of in-cluster evolution. This structure allows for efficient constant updates of the cluster structure, diminishing the influence of outdated records. The method is shown to outperform CluStream when applied to sliding windows and has bounded memory requirements. The EHCF is a flexible tool that can be used along other solutions.

6.2.1.3 Grid Clustering

In grid clustering, the d-dimensional input space is divided into cells by a set of (d-1)-dimensional hyperplanes. There exist multiple works that apply this type of algorithms to online settings. In [39], the authors describe methods for finding optimal hyperplanes according to certain criteria: hyperplanes should lie on low-density areas to avoid splitting clusters and they should discriminate clusters as much as possible. Another interesting approach using this notion is *Fractal Clustering*, that clusters points based on their impact on the fractal dimension of the group. In [47], D-Stream is presented. This algorithm makes a different partition for each dimension of the input space, which yields as many density grids as the product of the number of partitions for each dimension. Incoming points are mapped onto one of these grids and clusters are created offline based on the density of each grid. D-Stream was later extended with D-Stream II [48], which alleviates the problem of wrongly created clusters by considering the position of data points to decide whether to merge neighboring dense partitions or not. In [49] the authors describe MR-Stream, which according to the authors performs the online computations in constant time, allows for the discovery of clusters at multiple resolutions using a tree-like structure, determines the right time to generate clusters, generates highly pure clusters and determines the right threshold for density-based clustering. The tree built by the algorithm is pruned online. Other proposals include ExCC [50], DUCStream [51], DD-Stream [52], PKS-Stream [53] and DENGRIS-Stream [54].

6.2.1.4 Density-Based Clustering

Several density-based algorithms for streaming data have been proposed in recent literature. Among the most notable ones is DenStream [40]. This algorithm combines DBSCAN and the concept of micro-clustering on a fading window over a data stream, and bounds storage using a pruning procedure. It succeeds at recognizing arbitrarily shaped clusters. In [55], C-DenStream is described. This algorithm extends DenStream relying on certain external constraints to guide the clustering process. In particular, they utilize the notions of *Must-link* and *Cannot-link* to enforce or prevent particular groupings among the data based on prior knowledge. In [56], the authors propose HDDStream, with which they try to solve the problem of high-dimensional stream clustering, which can be challenging due to the presence of irrelevant features and the locality of relevance. It improves older partition-based HPStream [57], which assumes that the number of clusters does not change. It functions by summarizing sets of objects in their relevant dimensions. Finally, Flockstream [58] is a multi-agent flocking model. Its greatest advantage is that it does not need the offline component that other density-based methods utilize for the final clustering. Other proposals include StreamOptics [59], rDenStream [60], SDStream [61], HDenStream [62], SOStream [63] and PreDeConStream [64].

6.2.2 Network Traffic Prediction

In order to harness the complexity of network traffic evolution over time to make predictions it is essential to identify its dynamics, regularities and temporal interdependence. In the recent literature, there exist several proposals that employ machine learning techniques to enhance existing time series analysis and prediction methods or build new ones. Some of the works that have developed these methods in the field of network traffic and similar areas are described below.

6.2.2.1 Support Vector Regression

Some authors propose to use Support Vector Regression (SVR) as a method for the prediction of various network traffic variables. Support Vector Machines (SVM) are a very successful classification method that pose the problem of finding an optimal hyperplane to separate two sets of data points. It can be used for regression by imposing a maximum

distance constraint on the objective function. Although SVMs are a supervised approach, SVR does not need expert intervention, since the 'labels' in this case are the values of the input function. In [26], the authors use this technique to try to predict TCP throughput using different network metrics such as Available Bandwidth, Queue loss and Packet loss, and find the last two to be the most relevant. They claim to obtain measures within 10% of the actual value 87% of the time, although the data set used was artificially generated on their local network. In [27], another SVR based proposal is described. The authors of this work attempt to predict the link load based on previous records. The data set used is a one-day long capture from a major Italian ISP. They conclude that, even though SVR does not seem to yield significant improvements over ARMA models in terms of accuracy for short-term link load prediction, it presents extremely advantageous qualities such as robustness to parameter variation, low computational complexity and the possibility to extend the forecasting horizon by using cascading SVR models.

6.2.2.2 Dimensionality Reduction

The application of forecasting techniques can be especially challenging when dealing with high-dimensional input data spaces. This problem can be addressed using dimensionality reduction techniques such as Singular Value Decomposition (SVD) and PCA. In [28], SVD is used on YouTube video access data in order to extract patterns and make predictions more efficiently using an ARMA model. To perform predictions, they consider a data matrix comprised of 366 columns, each of which being a year-long time series of daily views for each video considered.

An ARMA model is built for the first principal components of this data matrix (matrix *U* of decomposition *USV*). The predicted vectors are then projected back onto the original space by multiplying them by matrices *S* and *V* for interpretation. Their results are encouraging, and the approach is sufficiently generic to be directly applied to network traces using certain features of interest, e.g. number of flows in the network, failed flow ratio, etc. In order to cope with rarely accessed videos, the authors use hierarchical clustering to group similar series. Predictions for a video are made mapping that video to a cluster and rescaling its mean time series.

6.2.2.3 Signal Processing & Neural Networks

Neural Networks have seen a significant resurgence since the potential of deep architectures started to be acknowledged and fast methods for training them were developed after 2006 [34][35]. These models have been applied to the problem of network traffic prediction combined with signal processing techniques. In [37], the authors build a multiresolution finite impulse response (FIR) neural network that uses a discrete wavelet transform (DWT) to adequately represent the time series data. The model is tested on the Bellcore Morristown Research and Engineering Center Ethernet traffic traces, and is shown to outperform an RLS predictor and a single resolution FIRNN. The slow convergence of the backpropagation algorithm, however, could hamper its applicability to online environments. In [31] the authors propose an optimized training procedure for a Bilinear Recurrent Neural Network (BLRNN) [36] and use it to predict the behavior of network traffic. The optimization consists of two parts: first, the multiplications for bilinear components are reduced; then, the network is pruned using a genetic algorithm. The method is also tested on the Bellcore data set. It shows improvements over traditional neural networks, but fails to predict peaks and in terms of accuracy it performs worse than the multi-resolution model described in [37]. However, its reduced computational cost is very relevant to online implementation.

7. Online traffic characterization architecture basics

7.1 Generic architecture design principle

This section aims at defining the principles for the generic architecture of the online classification system. We recall that WP4 aims at defining a scalable online network traffic characterization system.

Note also that WP4 architecture design principles take into account that WP5 Use Cases will have to integrate specific mechanisms developed in WP4.

One of the key functions to be designed and developed is related to scalable online flow classification. Based on this function, the two objectives are (1) to design a traffic pattern evolution sub-system, and (2) to design a network anomalies and intrusion detection sub-system (use case #1).

Given the online and scalable nature of the system and sub-systems, the key issues to be addressed are related to:

1. The traffic capture system and the way the capture traffic stream is afterwards sent to the flow classification system. This capture system has to be fast and needs to process data to transform it in the appropriate time series whose format has to be completed in the following months;
2. The computing of this large amount of data of high dimensions by the data-mining and Machine-Learning algorithms. These algorithms have to be able to handle large amounts of data in real time and in a scalable way.

7.1.1 Traffic capture

Figure 1 illustrates what has to be done to transform raw packet traces into aggregated traffic traces to be presented at the different data-mining and machine-learning algorithms that will be designed / used for flow classification, traffic pattern evolution, and anomaly detection. This architecture is aimed at being generic enough for coping with all possible algorithms that could run on top of it. It is especially possible this traffic aggregation task to be run on the capturing machines as they could perform it much faster than the machines running the machine learning and data mining algorithms. It that later case, it could lead to performance and scalability issues. The performance evaluation to be run in the second year of the project should provide the required figures for making the decision. Nevertheless, it will not change the principle of the proposed functional architecture.

Figure 1: Traffic pre-processing for scalable online characterization

7.1.2 Data processing step using data-mining and machine learning algorithms

Given the real-time and scalability objectives for the classification, traffic pattern study, and anomaly detection functions, the related algorithms need to be fast and efficient, in addition of being robust and accurate. The two next sections will detail the specific functional architecture for (1) traffic pattern evolution, and (2) anomaly detection.

7.2 Specific architecture details for traffic pattern evolution

In this section, an overview of the architectural foundations for a Traffic Pattern Evolution Subsystem is described. Some of the specific requirements and desirable properties of online algorithms for data streams are described in section 8.1 . All families of machine learning algorithms (classification, clustering, regression/prediction) can be applied to the problem of temporal traffic pattern evolution detection and analysis. However, some common ground for the different techniques that can be employed for this purpose can be established, and an abstract layout of the elements to participate in the traffic pattern evolution subsystem can be extracted.

7.2.1 Traffic Pattern Evolution Subsystem Architecture Description

As every other element of the ONTIC architecture, the Traffic Pattern Evolution Subsystem will process network data. These data must undergo a feature engineering process via a **Feature Engineering Subsystem** (FES) so that the algorithms involved can receive the data in an adequate representation.

Online streaming algorithms can consume data in various forms. Samples can be processed and immediately discarded one at a time. Alternatively, a *sliding window* can be maintained. In this case, the algorithm will need to keep the most recent sample (x_t) and the *n* previous ones (x_{t-n}, …, x_{t-1}) in memory in order to be able to utilize a *n+1* long window of data.

The **Scalable Stream Processing System** (Scalable-SPS) consists of the **Streaming Algorithm** (SA) or set of streaming algorithms to be used (please refer to section 6.2 for details), which will be deployed on top of an **Online Distributed Computing Framework** such as STORM or SAMZA. The adaptation of existing state-of-the-art techniques to these platform is one of the core goals of this project. Therefore, the Scalable-SPS will be one of the major contributions by ONTIC.

As described in section 6.2 , it is common for streaming machine learning algorithms to be comprised of an online and an offline component. The offline component can provide support for less time-critical tasks that can improve the online performance. For instance, a Support Vector Regression-based prediction system could immensely benefit from separate equipment for periodical retraining. The samples can be temporarily stored in a separate **Temporary Storage** device. When a temporal or storage threshold is surpassed, a **Batch Learner** can retrain the algorithm, dismiss the stored samples and transmit the new parameters to the predictor.

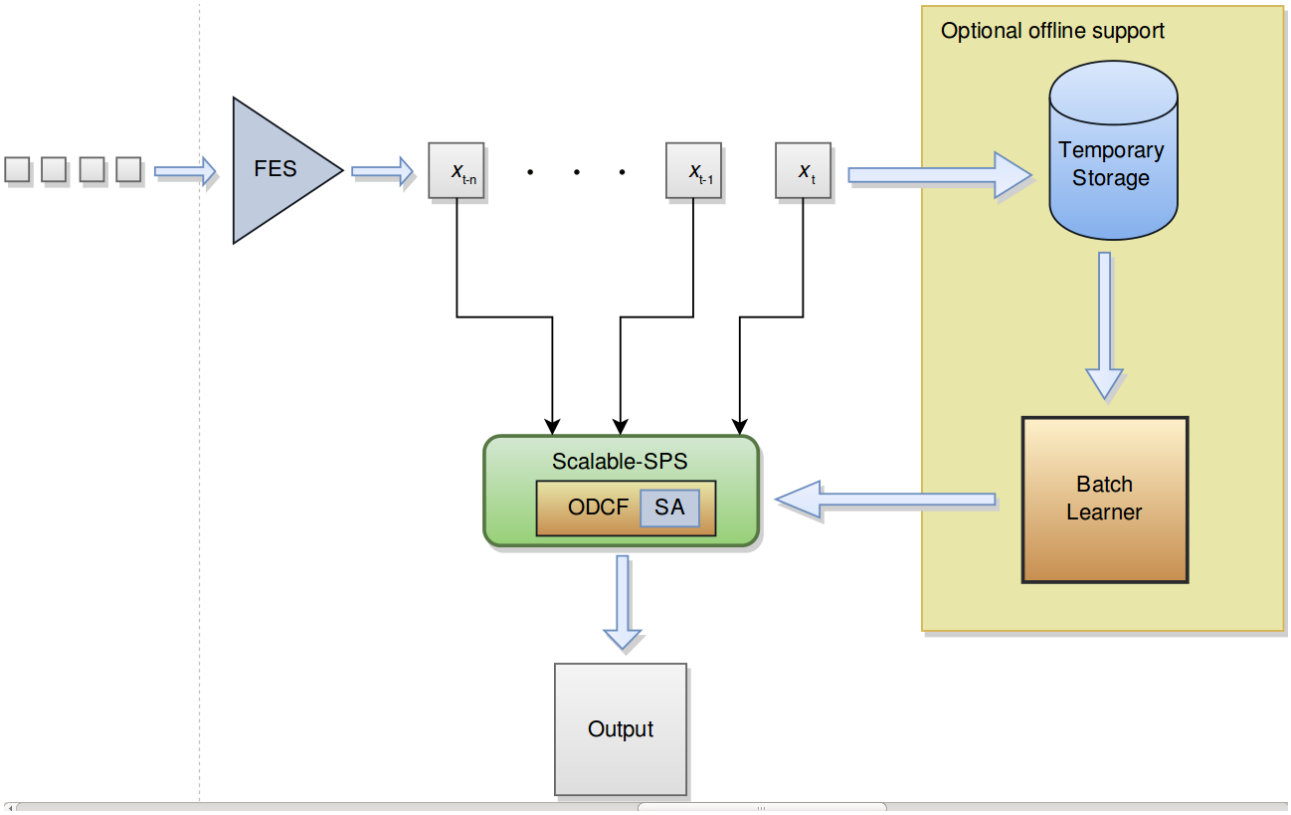

Figure 2: Traffic Pattern Evolution Subsystem Architecture

The Traffic Pattern Evolution Subsystem permanently generates output according to the problem at hand. A classifier determines the traffic class (e.g. application class) a packet, a set of packets, a flow or a set of flows belongs to. A clustering algorithm yields the description of the resulting clusters, which can contain the centroid location, the cluster density, the average cluster age and more. A forecasting system outputs a series of discrete forecasts, whose length depends on the established horizon. These problems pose challenges that fall within the scope of the ONTIC project. These challenges are discussed in section 8.1 .

7.3 Specific architecture details for anomaly detection

Most of the solution quoted in the state of the art (section 6.1) on anomaly detection (except unsupervised ones) share a common downside: they require the knowledge provided by an external agent to achieve their goal, either in terms of anomaly or attack signatures or as normal-operation profiles. As such, current network security and management look more like a reactive countermeasure than a proactive prevention mechanism. Over the past years we have, however, witnessed an increased interest within the network community in shifting away from reactive management and defense towards more proactive systems [21]. Our thesis behind this work is that reactive, knowledge-based approaches are not sufficient to tackle the network management and security problem, and that a holistic solution should also include proactive, knowledge-independent analysis techniques.

Armed with these ideas in mind, we present an Unsupervised Network Intrusion Detection System (UNADS) capable of detecting network anomalies without relying on signatures, training, or labeled traffic instances of any kind. Based on the observation that network anomalies, and particularly the ones that are the most difficult to detect, are contained in a small fraction of traffic instances with respect to normal-operation traffic [22] (we show that this hypothesis can

always be verified by using traffic aggregation), their unsupervised detection consists in identifying outliers, i.e. instances that are remarkably different from the majority. UNADS relies on robust clustering techniques to blindly extract the traffic instances that compose an anomaly. This unsupervised system runs in three consecutive steps, analyzing packets captured in contiguous time slots of fixed length. Figure 3: High-level description of the Unsupervised network Anomaly Detection System depicts a modular, high-level description of this system.

The first step consists in preparing the flow/features matrix that will represent the research space for the clustering algorithms. For doing so, captured packets are first aggregated into multiresolution traffic flows. The idea deals with considering the larger possible space including all features for the most possible accurate result. The scalability issue in terms of computing time will be addressed in next steps. Different time-series are then built on top of these flows.

The second step takes as input the full flow matrix. At this step, outlying flows are identified using a robust multi-clustering algorithm, based on a combination of Sub-Space Clustering (SSC) [18], Density-based Clustering [23], and Evidence Accumulation Clustering (EAC) [19] techniques. Based on the knowledge provided by this clustering algorithm, and an analytic formulae applied on network and traffic features (still to be improved), the system ranks the degree of abnormality of all the identified outlying flows, building an outlying flows ranking (see section 8.2 for more information).

In the third step, the top-ranked outlying flows are flagged as anomalies, using a simple threshold detection approach. Based on our first analysis in real case with previously labeled traffic as ground truth, threshold appears to be constant for discriminating between legitimate, erroneous and illegitimate traffic. Further studies will nevertheless been run in the coming months to confirm this first result.

The main contribution of UNADS relies on its ability to detect unknown attacks in a completely unsupervised fashion, avoiding the need of signatures, training, or labeled traffic flows. In addition, previous work [24] shows that evaluations on the computational time of UNADS permits to envisage an on-line operation of the system, using a parallel computing architecture for the core algorithms. This paper represents a continuation of our previous work on unsupervised anomaly detection [24].

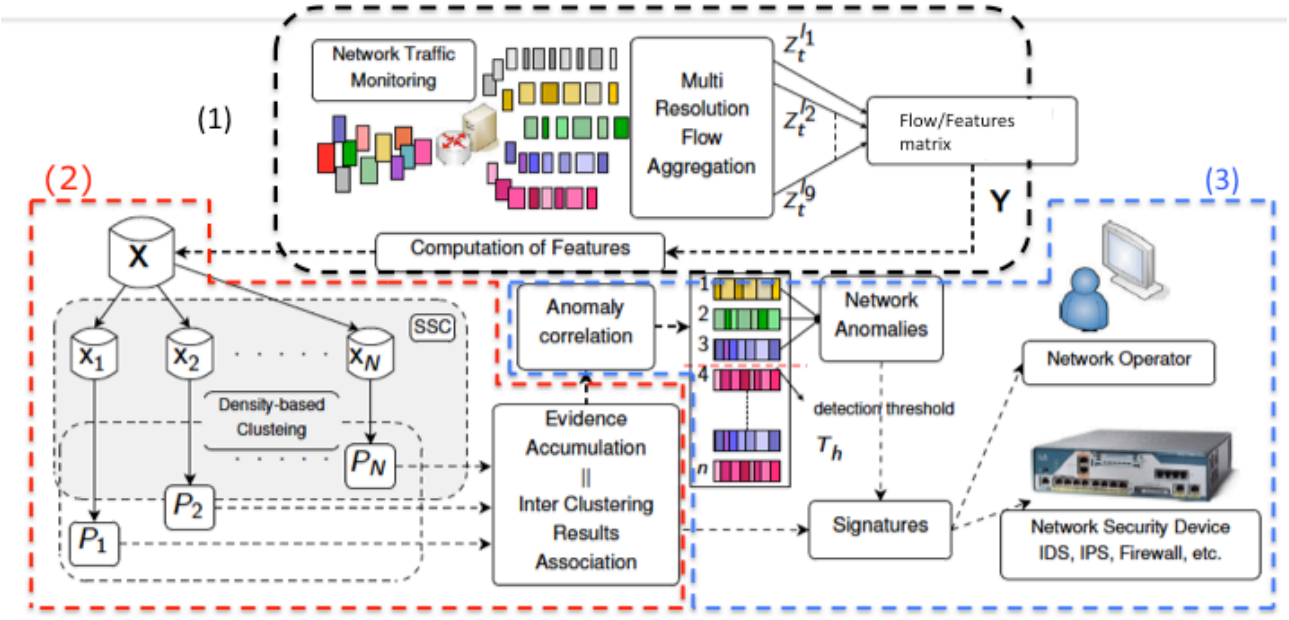

Figure 3: High-level description of the Unsupervised network Anomaly Detection System

8. Algorithm design basics and objectives

8.1 Traffic pattern evolution

Conventional machine learning algorithms require the complete data set to be available for training. This is known as batch processing. On the contrary, online streaming machine learning algorithms operate assuming that data streams are infinite, and can thus only function using a relatively small fraction of the data. Some of the ideal requirements for online algorithms are described below.

- Requirement 1: Process one sample at a time, at most once: Data streams are often continuous, which requires a stream mining algorithm to outpace their frequency. Due to the infinite nature of the stream, data samples must be processed and eventually discarded.
- Requirement 2: Use a limited amount of memory: a streaming algorithm must function in a manner such that the amount of memory needed is independently bounded.
- Requirement 3: Work in a limited amount of time: ideally, the running time complexity of a streaming algorithm will depend linearly on the number of treated samples. As mentioned before, stream mining algorithms must outpace the arrival rate of the data.
- Requirement 4: Readiness: These algorithms should ideally be ready to make predictions or decisions at any moment.

Of course, these requirements describe the ideal stream mining algorithm, but depending on the setting, some or all of them can be relaxed while maintaining a sufficiently satisfactory performance. The first requirement, for instance, can be overlooked if the algorithm can function using only a subset of the arriving samples. If some of the data are correctly discarded, the computational demands of the system can be significantly decreased. Requirement 2 can be relaxed as well. An exception can be made if external storage is used, although this can inflict unacceptable penalties on speed requirements. In general, these requisites can be observed with different levels of rigor if the environment allows it. The main restriction to be kept in mind by the algorithm designer is that the system must deal with a theoretically infinite amount of information.

Sometimes, the greatest benefits can be obtained from hybrid models, i.e. algorithms that combine batch processing with stream mining. Most machine learning-based prediction systems available in the literature are a good example of this. In the case of neural network-based predictors, for instance, the algorithms are capable of making predictions based on a window of time series values, but they must be trained offline. In general, these systems can be periodically retrained with the most recently collected data, and the newly found parameters can be transmitted to the predictor.

In addition to the requirements described above, online algorithms must deal with the fact that incoming data is often non-stationary. This is reflected in the notions of **concept-drift** (data evolve over time) and **concept-shift** (data change abruptly). These algorithms must also be robust to long-term deployment. A fading coefficient must dampen the influence of outdated samples on the performance of the system if necessary.

Classification and forecasting algorithms present specific challenges. Existing techniques can make decisions very fast. The decision function often runs in constant time, and techniques such

as Support Vector Machines, which are regarded as some of the slowest solutions, depend linearly on the number of support vectors (usually a very small subset of the training data).

Therefore, the challenge lies in the training process. In an online scenario, data are continuously generated (they are theoretically unlimited) and their nature can change over time (concept drift and concept shift). A trained model might become rapidly outdated, rendering the output of the algorithm unreliable. In order to address this, incoming data could be relayed to a training module to periodically update the model. However, training is usually very slow, and in big data scenarios this module would be almost immediately outpaced by the arriving data. Hence, a set of specific questions arises regarding this issue:

- Can we find fast enough alternatives or approximations for training algorithms to keep some state-of-the-art classification technique up to date in an online big data scenario?
- How can we determine how often we need to retrain the system?
- Can we summarize the data so that we can retrain the system without sacrificing samples?
- How much data do we need to use for training in order to have a sufficiently general model?

All these questions fall well within the scope of ONTIC. In the case of classification, however, the problem of labeling the data makes online solutions unfeasible. This reality makes it especially pertinent to lean toward unsupervised solutions.

8.2 Anomaly detection

In this deliverable we present a completely unsupervised method to detect and characterize network anomalies and attacks, without relying on signatures, training, or labeled traffic of any kind. Our approach relies on robust clustering algorithms to detect both well-known as well as completely unknown attacks, and to automatically produce easy-to-interpret signatures to characterize them, both in an on-line basis.

8.2.1 Traffic flow aggregation

The analysis is performed on packet-level traffic, captured in consecutive time slots of fixed length ΔT and aggregated in IP flows (standard *5-tuples*). IP flows are additionally aggregated at 9 different *flow* levels l_i. These include (from finer to coarser-grained resolution): *source IPs* (l_1: IPsrc}), *destination IPs* (l_2: IPdst), *source Network Prefixes* ($l_{3,4,5}$: IPsrc/24, /16, /8), *destination Network Prefixes* ($l_{6,7,8}$: IPdst/24, /16, /8}), and *traffic per Time Slot* (l_9: tpTS}).

8.2.2 Unsupervised detection of anomalies

The unsupervised detection stage takes as input all the IP flows in the considered time slot, aggregated according to one of the different aggregation levels used in the first stage. Let $\mathbf{Y} = \{y_1,.., y_n\}$ be the set of n flows in the considered time slot. Each flow $y_i \in \mathbf{Y}$ is described by a set of m traffic attributes or *features* on which the analysis is performed. The selection of these features is a key issue to any anomaly detection algorithm, and it becomes critical in the case of unsupervised detection, because there is no additional information to select the most relevant set. In this deliverable we shall limit our study to detect and characterize well-known anomalies and attacks, using a set of standard traffic features widely used in the literature. However, the reader should note that the approach can be easily extended to detect other types of anomalies

or attacks, considering different sets of traffic features. In fact, more features can be added to any standard list to improve detection and characterization results.

The set that we shall use here includes the following m = 9 traffic features: number of source/destination IP addresses and ports, ratio of number of sources to number of destinations, packet rate, ratio of packets to number of destinations, and fraction of ICMP and SYN packets. According to previous work on signature-based anomaly characterization [14], such simple traffic descriptors permit to describe standard network attacks such as DoS, DDoS, scans, and spreading worms/virus. Let $x_i = (x_i(1),.., x_i(m)) \in R^m$ be the corresponding vector of traffic features describing flow y_i, and $\mathbf{X} = \{x_1,.., x_n\}$ the complete matrix of features, referred to as the *feature space*.

The algorithm is based on clustering techniques applied to **X**. The objective of clustering is to partition a set of unlabeled elements into homogeneous groups of similar characteristics, based on some measure of similarity. Our goal is to identify in **Y** the different aggregated flows that may compose the attack. For doing so, the reader should note that an attack may consist of either outliers (i.e., single isolated flows) or compact small-size clusters, depending on the aggregation level of flows in **Y**. For example, a DDoS attack is represented as an outlier flow if the aggregation is done for IPdst, consisting of all the attacking IP flows sent towards the same victim. On the contrary, the attack is represented as a cluster if we use IPsrc flow-resolution. To avoid the lack of robustness of general clustering techniques, we have developed a parallel-multi-clustering approach, combining the notions of Density-based Clustering [23], Sub-Space Clustering [18], and Evidence Accumulation [19]. In what follows, we shall present the general idea behind the approach.

Instead of directly partitioning the complete feature space **X** using a traditional inter-flow similarity measure (i.e., the Euclidean distance), we do parallel clustering in N different sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ of smaller dimensions, obtaining N different partitions P_i of the flows in **Y**. Each sub-space $\mathbf{X}_i$ is constructed using only $r < m$ traffic features; this permits to analyze the structure of **X** from $N(m,r)$ different perspectives, using a finer-grained resolution. In particular, we do clustering in very-low dimensional sub-spaces, using r=2. To deeply explore the complete feature space, we analyze all the r-combinations-obtained-from-m sub-spaces; hence, $N(m) = m(m-1)/2$. The information provided by the multiple partitions P_i is then combined to produce a new similarity measure between the flows in **Y**, which has the paramount advantage of clearly highlighting both those outliers and small-size clusters that were simultaneously identified in different sub-spaces. This new similarity measure is finally used to easily extract the anomalous flows from the rest of the traffic.

8.2.3 Automatic characterization of anomalies

The following task after the detection of a group of anomalous flows is to automatically produce a set of K filtering rules $f_k(\mathbf{Y})$, $k=1,..,K$ to characterize them. In the one hand, such filtering rules provide useful insights on the nature of the anomaly, easing the analysis task of the network operator. On the other hand, different rules can be combined to construct a signature of the anomaly, which can be used to easily detect its occurrence in the future. To produce filtering rules $f_k(\mathbf{Y})$, the algorithm selects those sub-spaces $\mathbf{X}_i$ where the separation between the anomalous flows and the rest of the traffic is the biggest. We define two different classes of filtering rule: *absolute* rules $f_A(\mathbf{Y})$ and *relative* rules $f_R(\mathbf{Y})$. Absolute rules are only used in the characterization of small-size clusters, and correspond to the presence of dominant features in the flows of the anomalous cluster. An absolute rule for feature j has the form $f_A(\mathbf{Y}) = \{y_i \in \mathbf{Y}: x_i(j) == \lambda\}$. For example, in the case of an ICMP flooding attack, the vast majority of the

associated flows use only ICMP packets, hence the absolute filtering rule {nICMP/nPkts == 1} makes sense, (nICMP/nPkts corresponds to the fraction of ICMP packets).

On the other hand, relative filtering rules depend on the relative separation between anomalous and normal-operation flows. Basically, if the anomalous flows are well separated from the rest of the traffic in a certain partition P_i, then the features of the corresponding sub-space $\boldsymbol{X}_i$ are good candidates to define a relative filtering rule. A relative rule defined for feature j has the form $f_R(\boldsymbol{Y}) = \{y_i \in \mathbf{Y}: x_i(j) < \lambda \text{ or } x_i(j) > \lambda\}$. We shall also define a *covering relation* between filtering rules: we say that rule f_1 *covers* rule $f_2 \Leftrightarrow f_2(\boldsymbol{Y}) \subset f_1(\boldsymbol{Y})$. If two or more rules overlap (i.e., they are associated to the same feature), the algorithm keeps the one that covers the rest.

In order to construct a compact signature of the anomaly, we have to devise a procedure to select the most discriminant filtering rules. Absolute rules are important, because they define inherent characteristics of the anomaly. Regarding relatives rules, their relevance is directly tied to the degree of separation between flows. In the case of outliers, we select the K features for which the normalized distance to the normal-operation traffic (statistically represented by the biggest cluster in each sub-space) is among the top-K biggest distances. In the case of small-size clusters, we rank the degree of separation to the rest of the clusters using the well-known Fisher Score (FS) [25], and select the top-K ranked rules. The FS basically measures the separation between clusters, relative to the total variance within each cluster. To finally construct the signature, the absolute rules and the top-K relative rules are combined into a single inclusive predicate, using the covering relation in case of overlapping rules.

8.2.4 Experimental evaluation

This section is aimed at showing how the unsupervised approach can detect and characterize different network attacks without using signatures, labels, or learning.
We shall begin by detecting and characterizing a distributed SYN network scan directed to many victim hosts under the same */16* destination network. Packets in **Y** are aggregated using IPdst/24 flow resolution, thus the attack is detected as a small-size cluster. The length of each time slot is $\boldsymbol{\Delta T}$ = 20 seconds. As we explained in section 8.2.2 , the SSC-EA-based clustering algorithm constructs a new similarity measure between flows in **Y**, using the multiple clustering results obtained from the different sub-spaces. Let us express this new similarity measure as a $n \times n$ matrix S, in which element $S(i,j)$ represents the degree of similarity between flows i and j. Figure 4 depicts a histogram on the distribution of inter-flows similarity, according to S. The structure of flows in **Y** provided by S evidences the presence of a small isolated cluster in multiple sub-spaces. Selecting this cluster results in 53 anomalous IPdst/24 flows; a further analysis of the packets in these flows reveals multiple IP flows of SYN packets with the same IPsrc address and sequential IPdst addresses, scanning primary the same TCP port. Such a behavior is characteristic of a worm in the spreading phase.

Regarding filtering rules, Figure 5 depicts some of the partitions P_i where both absolute and top-K relative rules were produced. These involve the number of sources and destinations, and the fraction of SYN packets. Combining them produces a signature that can be expressed as $(nSrcs == 1) \wedge (nDsts > \lambda_1\}) \wedge (nSYN/nPkts > \lambda_2)$, where both λ_1 and λ_2 are obtained by separating clusters at half distance. Surprisingly enough, the extracted signature matches quite closely the standard signature used to detect such an attack in current signature-based systems [14]. The beauty and main advantage of the unsupervised approach relies on the fact that this new signature has been produced without any previous information about the attack or baseline traffic, and now it can be directly exported towards any security device to rapidly detect the same attack in the future.

Figure 6 depicts different rules obtained in the detection of a SYN DDoS attack. IP flows are now aggregated according to IPsrc resolution. The distribution analysis of inter-flows similarity with

respect to S selects a compact cluster with the most similar flows, corresponding to the set of attacking hosts. The obtained signature can be expressed as (nDsts == 1) ∧ (nSYN/nPkts > λ_3) ∧ (nPkts/sec > λ_4),} which combined with the large number of identified sources (nSrcs > λ_5) confirms the nature of a SYN DDoS attack. This signature is able to correctly isolate the most aggressive hosts of the DDoS attack, i.e., those with highest packet rate.

Figure 7 depicts the detection of an ICMP flooding DoS attack. Traffic is aggregated in IPdst flows. Thus the attack is now detected as an outlier rather than as a small-size cluster. Absolute rules are not applicable in the case of outliers detection. Relative rules correspond to the separation of the outlier from the biggest cluster in each sub-space, which statistically represents normal-operation traffic. Besides showing typical characteristics of this attack, such as a high packet rate of exclusively ICMP packets from the same source host, both partitions show that the detected attack does not involve the largest elephant flows in the time slot. This emphasizes the ability of the algorithm to detect attacks that are not necessarily different from normal-operation traffic in terms of volume, but that they differ in other, less evident characteristics. The obtained signature can be expressed as (nICMP/nPkts > λ_6) ∧ (nPkts/sec > λ_7).

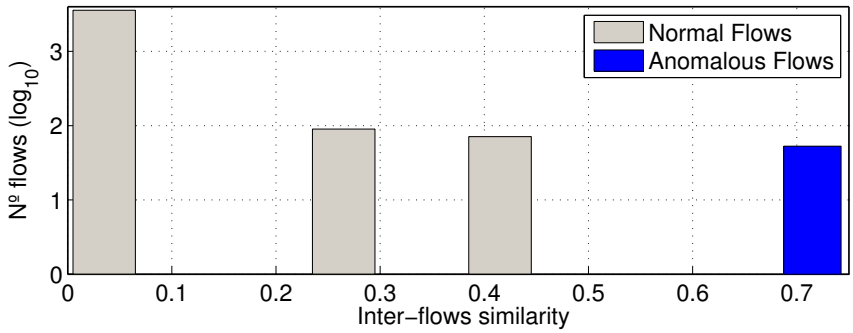

Figure 4: Detecting a distributed SYN network scan using S

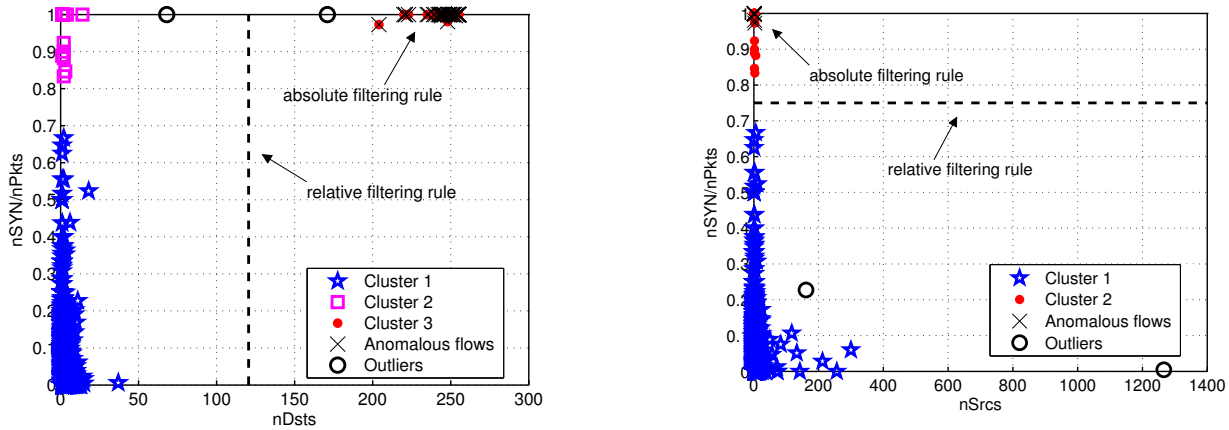

Figure 5: SYN network scan

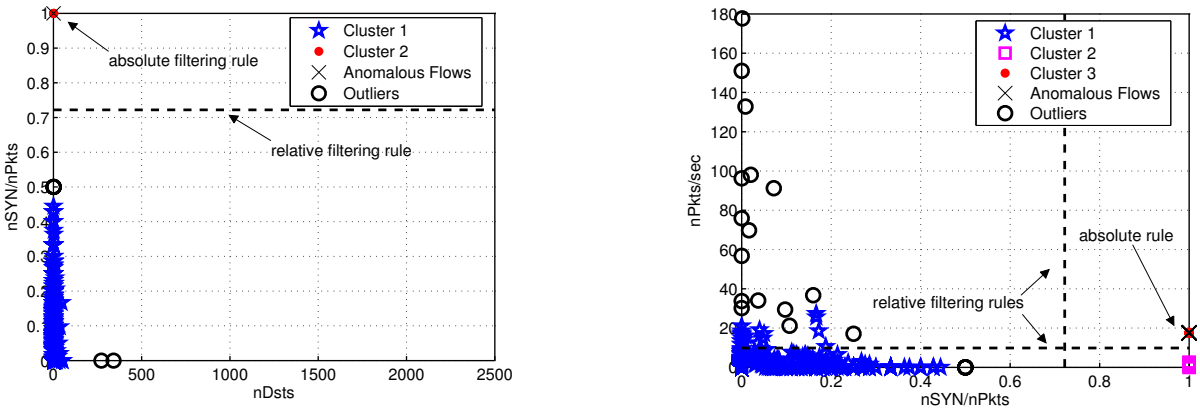

Figure 6: SYN DDoS

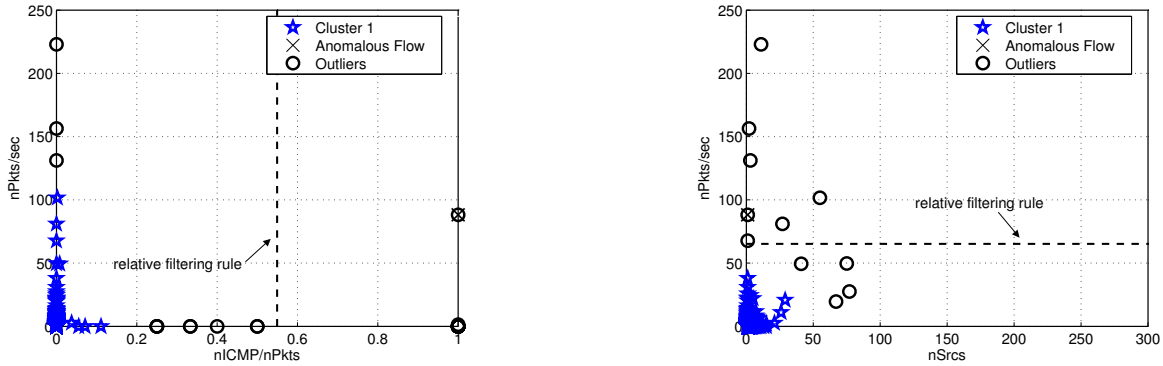

Figure 7: ICMP flooding DoS

8.2.5 Conclusion

The completely unsupervised algorithm for detection of network attacks that we have presented has many interesting advantages with respect to previous proposals. It uses exclusively unlabeled data to detect and characterize network attacks, without assuming any kind of signature, particular model, or canonical data distribution. This allows detection of new previously unseen network attacks, even without using statistical-learning. By combining the notions of Sub-Space Clustering and multiple Evidence Accumulation, the algorithm avoids the lack of robustness of general clustering approaches, improving the power of discrimination between normal-operation and anomalous traffic. We have shown how to use the algorithm to automatically construct signatures of network attacks without relying on any kind of previous information.

In the following month, we will continue improving the algorithm, perform intensive evaluation for estimating its accuracy and scalability, and start developing a prototype to be further included in the general ONTIC system.

9. References

[1] P. Barford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies,” Proc. ACM IMW, 2002.

[2] J. Brutlag, “Aberrant behavior detection in time series for network monitoring,” Proc. 14th Systems Administration Conference, 2000.

[3] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-based change detection: Methods, evaluation, and applications,” Proc. ACM IMC, 2003.

[4] A. Lakhina, M. Crovella, and C. Diot, “Characterization of network-wide anomalies in traffic flows,” Proc. ACM IMC, 2004.

[5] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” Proc. ACM SIGCOMM, 2005.

[6] A. Lakhina, C. Diot, and M. Crovella, “Diagnosing network-wide traffic anomalies,” Proc. ACM SIGCOMM, 2004.

[7] X. Li, F. Biang, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, “Detection and identification of network anomalies using sketch subspaces,” Proc. ACM IMC, 2006.

[8] P. Casas, S. Vaton, L. Fillatre, and I. Nikiforov, “Optimal volume anomaly detection and isolation in large-scale ip networks using coarse-grained measurements,” Computer Networks, vol. 54, pp. 1750-1766, 2010.

[9] H. Ringberg, A. Soule, J. Rexford, and C. Diot, “Sensitivity of PCA for traffic anomaly detection,” Proc. ACM SIGMETRICS, 2007.

[10] P. Casas, J. Mazel, and P. Owezarski, “Unada: Unsupervised network anomaly detection using sub-space outliers ranking,” IFIP Networking conference, 2011.

[11] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, “A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data,” Applications of Data Mining in Computer Security, Kluwer Publisher, 2002.

[12] K. Leung and C. Leckie, “Unsupervised anomaly detection in network intrusion detection using clustering,” Proc. ACSC05, 2005.

[13] L. Portnoy, E. Eskin, and S. Stolfo, “Intrusion detection with unlabeled data using clustering,” Proc. ACM DMSA Workshop, 2001.

[14] G. Fernandes and P. Owezarski, “Automated classification of network traffic anomalies,” Proc. SecureComm’09, 2009.

[15] F. Silveira and C. Diot, “RCA: Pulling anomalies by their root causes,” Proc. IEEE INFOCOM, 2010.

[16] G. Cormode and S. Muthukrishnan, “What’s new: Finding significant differences in network data streams,” IEEE Trans. on Networking, vol. 13 (6), pp. 1219-1232, 2005.

[17] J. Mazel, P. Casas, Y. Labit, and P. Owezarski, “Sub-space clustering, interclustering results association & anomaly correlation for unsupervised network anomaly detection,” 7th International Conference on Network and Service Management (CNSM 2011), CNSM’11, october 2011.

[18] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data: a review,” ACM SIGKDD Expl. Newsletter, vol. 6 (1), pp. 90-105, 2004.

[19] A. Fred and A. K. Jain, “Combining multiple clusterings using evidence accumulation,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 27 (6), pp. 835-850, 2005.

[20] A. K. Jain, “Data clustering: 50 years beyond k-means,” Pattern Recognition Letters, vol. 31 (8), pp. 651-666, 2010.

[21] N. Wyler, “Aggressive Network Self-Defense”, ISBN 978-1-931836-20-3, Syngress - Elsevier, 2005.

[22] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, “Network Anomaly Detection and Classification via Opportunistic Sampling”, in IEEE Network, vol. 23 (1), 2009.

[23] M. Ester, P. Kriegel, J. Sander, and X. Xu, “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, in Proc. ACM SIGKDD, 1996.

[24] J. Mazel, P. Casas, Y. Labit, P. Owezarski, “Sub-space clustering, inter-clustering results association & anomaly correlation for unsupervised network anomaly detection,” International Conference on Network and Service Management (CNSM’2011), Paris, France, 24-28 October 2011

[25] T. Jaakkola and D. Haussler, “Exploiting Generative Models in Discriminative Classifiers,” Advances in Neural Inf. Processing Sys. II, pp. 487-493, 1998.

[26] M. Mirza, J. Sommers, P. Barford, and X. Zhu, “A machine learning approach to tcp throughput prediction,” in ACM SIGMETRICS Performance Evaluation Review, vol. 35, no. 1. ACM, 2007, pp. 97–108.

[27] P. Bermolen and D. Rossi, “Support vector regression for link load prediction,” Computer Networks, vol. 53, no. 2, pp. 191–201, 2009.

[28] G. Gursun, M. Crovella, and I. Matta, “Describing and forecasting video access patterns,” in INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 16–20.

[29] S. Di, D. Kondo, and W. Cirne, “Google hostload prediction based on bayesian model with optimized feature combination,” Journal of Parallel and Distributed Computing, vol. 74, no. 1, pp. 1820–1832, 2014.

[30] V. Hodge, R. Krishnan, T. Jackson, J. Austin, and J. Polak, “Short-term traffic prediction using a binary neural network,” in 43rd Annual UTSG Conference, Open University, Milton Keynes, UK, 2011.

[31] D.-C. Park, “Structure optimization of bilinear recurrent neural networks and its application to ethernet network traffic prediction,” Information Sciences, vol. 237, pp. 18–28, 2013.

[32] Y. Chen, B. Yang, and Q. Meng, “Small-time scale network traffic prediction based on flexible neural tree,” Applied Soft Computing, vol. 12, no. 1, pp. 274–279, 2012.

[33] S. Suthaharan, “Big data classification: Problems and challenges in network intrusion prediction with machine learning,” ACM SIGMETRICS Performance Evaluation Review, vol. 41, no. 4, pp. 70–73, 2014.

[34] G. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” Neural computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[35] Y. Bengio and Y. LeCun, “Scaling learning algorithms towards ai,” Large-scale kernel machines, vol. 34, pp. 1–41, 2007.

[36] D. C. Park and Y. Zhu, “Bilinear recurrent neural network,” in Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, vol. 3. IEEE, 1994, pp. 1459–1464.

[37] V. Alarcon-Aquino and J. A. Barria, “Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction,” Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 36, no. 2, pp. 208–220, 2006.

[38] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams,” in Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, 2003, pp. 81–92.

[39] A. Hinneburg, D. A. Keim et al., “Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering,” in VLDB, vol. 99. Citeseer, 1999, pp. 506–517.

[40] F. Cao, M. Ester, W. Qian, and A. Zhou, “Density-based clustering over an evolving data stream with noise.” in SDM, vol. 6. SIAM, 2006, pp. 326–337.

[41] P. S. Bradley, U. M. Fayyad, C. Reina et al., “Scaling clustering algorithms to large databases.” in KDD, 1998, pp. 9–15.

[42] F. Farnstrom, J. Lewis, and C. Elkan, “Scalability for clustering algorithms revisited,” ACM SIGKDD Explorations Newsletter, vol. 2, no. 1, pp. 51–57, 2000.

[43] L. O’callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha, “Streaming-data algorithms for high-quality clustering,” in 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE Computer Society, 2002, pp. 0685–0685.

[44] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, “Streamkm++: A clustering algorithm for data streams,” Journal of Experimental Algorithmics (JEA), vol. 17, pp. 2–4, 2012.

[45] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O’Callaghan, “Clustering data streams: Theory and practice,” Knowledge and Data Engineering, IEEE Transactions on, vol. 15, no. 3, pp. 515–528, 2003.

[46] A. Zhou, F. Cao, W. Qian, and C. Jin, “Tracking clusters in evolving data streams over sliding windows,” Knowledge and Information Systems, vol. 15, no. 2, pp. 181–214, 2008.

[47] Y. Chen and L. Tu, “Density-based clustering for real-time stream data,” in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007, pp. 133–142.

[48] L. Tu and Y. Chen, “Stream data clustering based on grid density and attraction,” ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 3, no. 3, p. 12, 2009.

[49] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, “Density-based clustering of data streams at multiple resolutions,” ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 3, no. 3, p. 14, 2009.

[50] V. Bhatnagar, S. Kaur, and S. Chakravarthy, “Clustering data streams using grid-based synopsis,” Knowledge and Information Systems, pp. 1–26, 2013.

[51] J. Gao, J. Li, Z. Zhang, and P.-N. Tan, “An incremental data stream clustering algorithm based on dense units detection,” in Advances in Knowledge Discovery and Data Mining. Springer, 2005, pp. 420–425.

[52] C. Jia, C. Tan, and A. Yong, “A grid and density-based clustering algorithm for processing data stream,” in Genetic and Evolutionary Computing, 2008. WGEC’08. Second International Conference on. IEEE, 2008, pp. 517–521.

[53] J. Ren, B. Cai, and C. Hu, “Clustering over data streams based on grid density and index tree,” Journal of Convergence Information Technology, vol. 6, no. 1, 2011.

[54] A. Amini and W. T. Ying, “Dengris-stream: A density-grid based clustering algorithm for

evolving data streams over sliding window," in International Conference on Data Mining and Computer Engineering (ICDMCE), 2012, pp. 206–210.

[55] C. Ruiz, E. Menasalvas, and M. Spiliopoulou, "C-denstream: Using domain knowledge on a data stream," in Discovery Science. Springer, 2009, pp. 287–301.

[56] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kr¨oger, and H.-P. Kriegel, "Density-based projected clustering over high dimensional data streams." in SDM. SIAM, 2012, pp. 987–998.

[57] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004, pp. 852–863.

[58] A. Forestiero, C. Pizzuti, and G. Spezzano, "A single pass algorithm for clustering evolving data streams based on swarm intelligence," Data Mining and Knowledge Discovery, vol. 26, no. 1, pp. 1–26, 2013.

[59] D. K. Tasoulis, G. Ross, and N. M. Adams, "Visualising the cluster structure of data streams," in Advances in Intelligent Data Analysis VII. Springer, 2007, pp. 81–92.

[60] L. Li-xiong, K. Jing, G. Yun-fei, and H. Hai, "A three-step clustering algorithm over an evolving data stream," in Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, vol. 1. IEEE, 2009, pp. 160–164.

[61] J. Ren and R. Ma, "Density-based data streams clustering over sliding windows," in Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on, vol. 5. IEEE, 2009, pp. 248–252.

[62] J. Lin and H. Lin, "A density-based clustering over evolving heterogeneous data stream," in Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS Interna tional Colloquium on, vol. 4. IEEE, 2009, pp. 275–277.

[63] C. Isaksson, M. H. Dunham, and M. Hahsler, "Sostream: Self organizing density-based clustering over data stream," in Machine Learning and Data Mining in Pattern Recognition. Springer, 2012, pp. 264–278.

[64] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl, "Density-based projected clustering of data streams," in Scalable Uncertainty Management. Springer, 2012, pp. 311–324.

[65] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in ACM SIGMOD Record, vol. 25, no. 2. ACM, 1996, pp. 103–114.