Machine Learning in Software Defined Networks: Data Collection and Traffic Classification

Pedro Amaral
Instituto de Telecomunicações,
Dep.de Eng. Electrotécnica,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa
Email: pfa@fct.unl.pt

João Dinis
Dep.de Eng. Electrotécnica,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa
Email: jp.dinis@campus.fct.unl.pt

Paulo Pinto
Instituto de Telecomunicações,
Dep.de Eng. Electrotécnica,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa
Email: pfp@fct.unl.pt

Luis Bernardo
Instituto de Telecomunicações,
Dep.de Eng. Electrotécnica,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa,
Email: lflb@fct.unl.pt

João Tavares
Reditus Network Innovation
Email: jlt001@hro.reditus.pt

Henrique S. Mamede
Universidade Aberta,
INESC TEC,
Reditus Network Innovation,
Email: Henrique.Mamede@reditus.pt

***Abstract*—Software Defined Networks (SDNs) provides a separation between the control plane and the forwarding plane of networks. The software implementation of the control plane and the built in data collection mechanisms of the OpenFlow protocol promise to be excellent tools to implement Machine Learning (ML) network control applications. A first step in that direction is to understand the type of data that can be collected in SDNs and how information can be learned from that data. In this work we describe a simple architecture deployed in an enterprise network that gathers traffic data using the OpenFlow protocol. We present the data-sets that can be obtained and show how several ML techniques can be applied to it for traffic classification. The results indicate that high accuracy classification can be obtained with the data-sets using supervised learning.**

***Index Terms*—Software defined Networks; Machine Learning; Data Analysis; Traffic Classification**

I. Introduction

The quantity of data that flows through telecommunications networks and the complexity of the applications that generate it keeps rising. Extracting knowledge from this data to understand and predict its impact is becoming increasingly important to perform an efficient management of the network.

The new paradigm of *Software Defined Networks (SDNs)*, unlocks new paths in this area with its built in information gathering mechanisms and the flexibility and programmability that it has brought to networks. OpenFlow is the standard southbound API protocol for SDNs. It defines how to communicate between the controller plane, that runs in software, and the data plane hardware.

In this paper we explore what kind of knowledge is possible to extract from the information made available by OpenFlow about the data flowing in an SDN network and in what ways can this information be used to manage/control the network. This study is the first step towards an SDN application for the control of enterprise computer networks using data driven Machine Learning (ML) and is focused in the monitoring and classification of traffic using data obtained with the OpenFlow protocol.

There have been several approaches to try to identify the traffic flowing in a network. Traditionally transport layer port information and the direct inspection of packet's payload have been used to such purpose. However, both approaches have several shortcomings and *artificial intelligence (AI)* algorithms, more specifically, *machine learning (ML)*, ones, have been considered in more recent literature.

The possibilities that emerge from the usage of ML in this context are various, some examples are: Forecasting tendencies in the network (e.g. predicting traffic volumes); Security monitoring (e.g. access control, adaptive firewalls, and DoS and DDoS attack detection; Traffic Engineering (TE) with traffic identification and congestion prediction.

Software defined networks have the potential to ease the implementation of such applications in networks. They provide a separation between the data plane and the forwarding plane, with the control plane in software and the possibility to have a centralized vision of the network. This software control point and the built in data collection mechanisms facilitates the harvesting of the data and the implementation of ML algorithms with no need for extra middleware for that purpose and with the possibility to enforce network behaviors based on the results.

In this paper, we propose a simple architecture to collect traffic data in an enterprise network using OpenFlow that can be used in SDN networks as well as in legacy networks (in this case using a single OpenFlow switch). We then present

This work was supported by the LoCoSDN project financed by FCT/MEC Project IT PEst- UID/EEA/50008/2013

the obtained data set, the data mining tools used to process it and the initial results of applying ML techniques for traffic classification on the collected data.

II. Related Work and Motivation

Several Traffic classification techniques have been proposed and used in communication networks over the years.

A. Port based IP traffic classification

Port based classification was one of the most used techniques in the past, and it was successful, to some extent, because many applications used fixed port numbers assigned by *Internet Assigned Numbers Authority* (IANA). However, over time some limitations to this approach became obvious. A high number of applications appeared that do not have registered port numbers, and many of those use dynamic port-negotiation mechanisms to hide from firewalls and network security tools. Also, the use of IP layer encryption, obfuscation and proxies may also hide the TCP or UDP header, making it impossible to discover the original port numbers.

B. Payload based IP traffic classification

The payload approach also known as Deep Packet Inspection (DPI), is currently one of the most utilized techniques. DPI systems inspect the payload of the packets searching for patterns that identify the application. These patterns, also called signatures, are defined from regular expressions that are evaluated by automata that work in a sequential way requiring a big amount of memory. This can cause a scalability problem that is even more serious since DPI is performed in the communication path.

To reduce the scalability problem, several techniques have been developed along two different directions. One is the modification of the automata like Fast Finite Automata (FFA) [2], Deterministic finite Automata (DFA) [15], Delayed input DFA (D^2FA) [11] and Differential encoding of DFAs [6]. The other one is using specific hardware like *Field Programmed Gate Array* (FPGA) or more recently *Graphics Processing Units* (GPUs). The use of GPUs to do the processing is not a trivial task, because these units were made for *Single Instruction Multiple Threads* (SIMT), and the operation of the automata is sequential.

In addition to these problems, the devices that need to be introduced in the network have a high cost and increase the complexity of the network. DPI can also be very difficult or impossible when dealing with encrypted traffic which has been increasing. Finally, there are privacy concerns with examining user data.

C. Machine learning classification

Machine Learning (ML) techniques can also be used for traffic classification and several works have been proposed [13].

Traffic classification can be performed using supervised learning [10] with algorithms such as Support Vector Machines (SVM), neural networks and decision trees. In supervised learning there is a need to obtain labeled training data-sets, something that can be challenging in computer networks due to the difficulty in obtaining accurately annotated network flow samples across a broad range of applications and the rate at which new applications can appear. This results in many of this approaches being limited to coarse-grained classifications such as web, P2p and VoiP [9].

An alternative is the use of unsupervised ML where the data given to the learner is unlabeled. Unsupervised ML is normally used for clustering tasks, where the algorithms group the data into different clusters according to similarities in the feature values [17]. The goal is to find unknown relationships in the data, finding similarity patterns between the several observations. There are several algorithms that can be used in unsupervised ML and there is abundant literature on their application in traffic classification. Approaches based in K-Means or K-Medoids [1], [4], [12], [16], Self-Organizing Maps (SOM) [8], [14] and Density Based Spatial Clustering of Applications with Noise (DBSCAN) [4] have been proposed for several traffic classification scenarios.

A combination of unsupervised and supervised methods, refereed to as semi-supervised learning [5], can also be used with both labeled and unlabeled data. This approach tries to overcome the difficulties in obtaining labeled data. It can work with a data-set where the majority of the instances are unlabeled if a small number of labeled data instances exists that permits the mapping from the identified clusters in the overall data-set into the different classes.

D. Using SDN Data for ML Classification

Machine Learning approaches to traffic classification are usually applied to data that is obtained from dedicated data gathering protocols based in the Internet Protocol Flow Information eXport (IPFIX) standard. Such approaches are usually used in ISPs where routers work as measurement points to collect data for storage and processing. IPFIX aggregates packets into flows via aggregation rules and provides flow information like flow duration, flow bit rate and flow inter-packet arrival times. This information is then processed in a separate module for ML based traffic classification.

SDN provides new ways to perform these operations. OpenFlow has built in flow statistics and the forwarding rules (Flow Entries) installed in the switches match packets into flows at several possible granularities. There are also multiple levels at which such information can be collected, from access switches where the number of flows is smaller to core switches or gateway switches where a high number of traffic flows can exist.

Another difference is that statistics are received directly in the controller, where the control plane resides, opening the possibility to directly use learned information, like the classification of traffic, for network management. If the information on the traffic is learned on-line, consider as an example the classification of a traffic flow after a short initial number of packets, the controller can immediately act on that flow by applying new forwarding rules to its packets.

There is also a lot of flexibility in how data can be collected. It can be customized by defining different types of flow granularities for different types of traffic and a customizable number of packets can be analyzed in the controller before installing a flow rule in the switch to gather its statistics.

The motivation behind the work reported in this paper is the development of an SDN network management application for enterprise networks that uses machine learning. The first step towards this goal is to study how traffic information can be learned by an SDN controller using SDN mechanisms. We started by focusing on TCP traffic since the majority of applications of interest in an enterprise network are transported over TCP.

III. Architecture

We developed an SDN application that gathers OpenFlow statistics from the controlled switches. The application proactively uses OpenFlow to install a Flow_Entry [1] that instructs the controlled switches to forward all traffic to the controller.

When a packet arrives at the controller (inside an Openflow Packet_In message) the controller analyses it to determine its transport protocol. For TCP packets the TCP flags are analyzed to detect the packets involved in the initial TCP handshake. After the handshake the first five packets exchanged between the client and the server are still received in the controller. For each one the size and arrival time stamp is stored. At the fifth packet, the controller installs a Flow_Entry in the switch with an idle timeout and the rest of the packets that belong to that connection are processed locally in the switch until the idle timeout expires indicating that packets matching to that flow are no longer arriving at the switch. The use of a Flow_Entry is to reduce the number of Packet_In messages in the controller. If the application is only for traffic monitoring, the packet processing instruction at the flow entries can be for the switch to forward the packets as a regular switch (using the forward normal OpenFlow instruction) or, if the received traffic is a copy (for example receiving a mirrored port from a traditional switch), a drop action (an empty OpenFlow action). When a flow entry is removed due to timeout, its statistics, namely the packet count and byte count, are collected at the controller.

In this work the application only serves as a data collection application and it can be deployed in two different ways. In the first case, only possible in SDN networks, the controller is connected to the switches of the SDN network that might be points of data collection. The application then chooses the set of switches that will collect data and performs the above actions on those switches.

In the second case, that can also be applied in legacy non-SDN networks, we have a single OpenFlow enabled switch and use port mirroring to obtain a copy of the traffic of the measurement point delivered in a port of that switch. The type of traffic data that is obtained just depends on where the switches that provided it are located in the network (or where the port mirroring is performed in the second case). If they are access switches the controller can obtain information on the traffic from the hosts, while in core switches information of the traffic from the entire network is available.

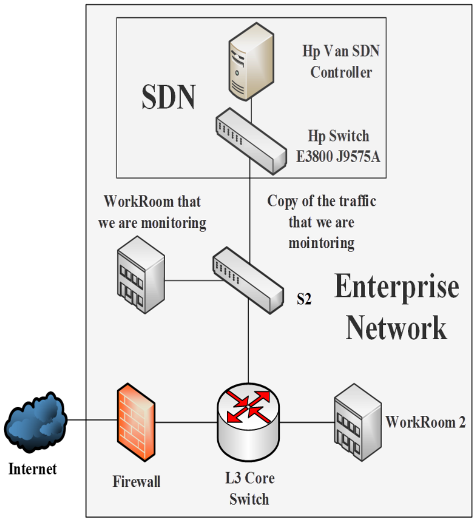

Fig. 1. Deployed Topology

A. Deployment

To obtain the data-set studied in this work we deployed a single OpenFlow Switch in a non-SDN enterprise production network as depicted in figure 1. The OpenFlow switch receives a copy of the traffic of the upstream link connecting the monitored workroom to the core router, that copy is provided via port mirroring at switch S2. We use an HP E3800 Openflow enabled switch and the HP VAN SDN controller to run the traffic monitoring application that uses OpenFlow version 1.3 to control the switch.

This setup is very lightweight and it does not interfere with the normal flow of the traffic in the monitored network.

We obtained two different data sets, an unlabeled data-set containing info from all traffic from the monitored room and a smaller data-set produced under controlled conditions and labeled with the corresponding applications that generated the data.

Labeled data is obtained by isolating the traffic of a single host in switch S2 (i.e one of the access ports is mirrored instead of mirroring the uplink port). By controlling the host generating the mirrored traffic we can label the data with the corresponding applications.

B. Collected Data

For all the TCP flows detected in the controller the SDN application stores, in a CSV file, the features shown in table I: the size, timestamps and inter-arrival time of the first five packets.; the MAC and IP source and destination addresses; the source and destination transport ports; the duration, the byte count and the packet count of the connection.

[1] Each flow entry contains packet match fields, priority, various counters, packet processing instructions, timeouts and a cookie.

TABLE I
STORED FEATURES FOR A TCP FLOW

Feature	Description
Packet Size (1 to 5)	Size of the first five packets payload in bytes
Packet Time stamp (1 to 5)	Arrival time of the first five packets in milliseconds (controller system time)
Inter-arrival time (first 5 packets)	Timestamp $Packet_i-$ Timestamp $Packet_i - 1$
SrcMAC	Source MAC address
DstMAC	Destination MAC address
SrcIP	Source IP address
DstIP	Destination IP address
SrcPort	Source Transport Port
DstPort	Destination Transport Port
Flow Duration	timestamp Flow_Removed - Timestamp $Packet_1$
Byte Count	Flow_Entry byte count statistics
Packet Count	Flow_Entry Packet count statistics

TCP flows are detected by inspecting the flags in the payload of the Packet_In OpenFlow messages received in the controller. The size and timestamps of the first five packets are also directly obtained by storing the size of the Packet_In payload and the arrival time of the messages. The inter-arrival times of the first five packets is then computed from that information.

The addresses and ports are also obtained via those initial Packet_In messages and are used to build the match clause for the OpenFlow Flow_Entry that is installed in the switch to deal with the rest of the packets of that TCP connection. The byte count and packet count are obtained via the statistics of the Flow_Entry. Finally the duration is obtained by subtracting the timestamp of the initial packets from the timestamp of the flow removed message received by the controller when the Flow_Entry timeout expires.

C. Data Processing

The collected data has very different features with values in different scales. In order to avoid features with large scale values to overlap smaller scale ones causing bad performance in the ML algorithms, the collected data is treated to scale and center the feature values. This is done by calculating the standard scores for each data feature. The standard score x', of a data feature x is given by:

$$x' = \frac{x - \overline{x}}{\sigma(x)}$$

Where $\sigma(x)$is the standard deviation and $\overline{x}$ the distribution mean value for x. Standardized features have approximately zero mean and unit standard deviation thus eliminating high variability and scaling effects.

Another important issue is the correlation between features. Highly correlated features are somewhat redundant and add little to the performance of the classifier algorithms. Eliminating such redundant features is important since it reduces the computation cost while maintaining the classification accuracy. A *Principal Component Analysis* (PCA) algorithm is used in the data set in order to find the principal components (the features that are linearly uncorrelated). This will reduce the number of features in cases where there are correlated data.

IV. RESULTS

The goal of this first work was to test the performance of supervised classification algorithms in this scenario. To do this we obtained a labeled data-set using a single host to generate the TCP connections. The traffic generated by the host is then sent via port mirroring to the OpenFlow switch where the SDN application in the controller performs the data collection. This results in a data-set where the application responsible for the features measured in each TCP connection is known. We collected 500 feature set samples for each of the following applications: Youtube, Vimeo, Facebook, Linkedin, Skype, Bittorrent, Web Browsing(HTTP) and Dropbox.

To test the classification accuracy, the labeled data-set must be divided into training and testing sets. We use a bootstrap estimate technique, using the R programming language, where n random instances are chosen from the data-set (that also contains n elements). A random number is generated in R with a seed number of 12345 and the corresponding line is chosen from the data-set. The repeated instances are ignored leaving only the unique ones that constitute the training set. The remaining values are used as the testing set.

We tested several ensemble learning classifiers, these classifiers use a set of regular classifiers and classify data by taking a weighed vote of each individual prediction. They perform better than each of the individual classifier algorithms that they use. More specifically we tested, Random Forests (RF) [3], and two variations of gradient boosting classifiers [7], Stochastic Gradient Boosting (SGB) and Extreme Gradient Boosting (EGB).

The classifiers are trained using the training set and evaluated against the testing set. We repeat the bootstrap, training and testing process 30 times for each algorithm and calculate the overall accuracy of the classifier as the average of the accuracy values in each test.

Accuracy is measured by comparing the labels yielded by the classifier to the true labels collected in the experimental setup explained previously. The results are shown in table II.

As we can see these initial results are promising with high accuracy values for all the tested applications. Results were very similar across the tested algorithms showing that the data obtained via SDN mechanisms is suitable for ML supervised traffic classification. There is off course the inherent difficulty of obtaining labeled sets in networks, however our experiment shows that in small campus or enterprise networks one can produce a small set of labeled data-sets for a few applications of interest for each particular scenario and classify traffic with high confidence with a very simple software running at the controller and a single switch involved.

TABLE II
CLASSIFIER ACCURACY

Application	Random Forests Accuracy (%)	Stochastic Gradient Boosting Accuracy (%)	Extreme Gradient Boosting Accuracy (%)
Bittorrent	91.20	89.60	90.40
Dropbox	92.80	89.60	90.40
Facebook	90.40	84.80	91.20
HTTP	96.00	93.60	95.20
Linkedin	73.60	73.60	76.80
Skype	90.40	90.40	91.20
Vimeo	73.60	71.20	73.60
Youtube	83.20	91.20	88.80

V. CONCLUSIONS AND FUTURE WORK

We present a traffic classification architecture based in SDN that allows the collection of traffic data both in SDN networks and in legacy networks. It is based solely on OpenFlow and can be implemented in an SDN app in the controller, paving the way for the use of data driven learning algorithms for network control. We obtained initial results that indicate that supervised learning algorithms can be used with this architecture and with the data that it collects with high accuracy levels. We also show that for small campus and enterprise networks this is a feasible approach to identify traffic of interest. As future work the study of unsupervised methods or semi-supervised methods can be explored for Traffic classification. The study of other types of useful information that can be learned from the collected data like, user usage profiles, network use profiles or traffic predictions is also a next step.

REFERENCES

[1] T. Bakhshi and B. Ghita. User traffic profiling. In *Internet Technologies and Applications (ITA), 2015*, pages 91–97. IEEE, 2015.

[2] M. Becchi and P. Crowley. An improved algorithm to accelerate regular expression evaluation. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, pages 145–154. ACM, 2007.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM, 2006.

[5] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64(9):1194–1213, 2007.

[6] D. Ficara, A. Di Pietro, S. Giordano, G. Procissi, F. Vitucci, and G. Antichi. Differential encoding of DFAs for fast regular expression matching. *IEEE/ACM Transactions On Networking*, 19(3):683–694, 2011.

[7] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[8] D. Jiang, Y. Yang, and M. Xia. Research on intrusion detection based on an improved SOM neural network. In *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, volume 1, pages 400–403. IEEE, 2009.

[9] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT conference*, page 11. ACM, 2008.

[10] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[11] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 339–350. ACM, 2006.

[12] Y. Liu, W. Li, and Y.-C. Li. Network traffic classification using k-means clustering. In *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*, pages 360–365. IEEE, 2007.

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

[14] V. Pachghare, P. Kulkarni, and D. M. Nikam. Intrusion detection system using self organizing maps. In *Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. International Conference on*, pages 1–5. IEEE, 2009.

[15] R. Smith, C. Estan, S. Jha, and S. Kong. Deflating the big bang: fast and scalable deep packet inspection with extended finite automata. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 207–218. ACM, 2008.

[16] M. Vandana and S. Manmadhan. Self learning network traffic classification. In *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, pages 1–5. IEEE, 2015.

[17] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Computer Communication Review*, 36(5):5–16, 2006.