A SDN Data Plane Classifier for Encrypted Traffic

Parth Shah, Emilio Roche, Richard Purcell, Ali Shan Khawaja
Nadish Maredia, Pengxiang Sun, Miguel Neves, Israat Haque
Dalhousie University
{parthshahk, emilio.roche, richard.purcell, al290543, Nadish.Maredia, pn404317}@dal.ca

***Abstract*—Traffic classification is an important concept in network management, assisting with performance monitoring, provisioning, capacity planning and maintaining security. It is also challenging, especially with the dynamic nature of web protocols, implementing encryption in new protocols such as Google's QUIC and older protocols such as HTTP/1 moving to HTTP/2. We propose an efficient traffic classifier to deal with these problems. As well, some past implementations of Machine Learning classification have relied on the host server's identity. When the header fields for these extensions, such as Server Name Indication (SNI) in the Transport Layer Security (TLS) extension are removed, the classifier fails. In this project, we create a decision tree classifier that is protocol agnostic. It differentiates traffic into classes, such as browsing, chat, email, streaming, file transfer, and VOIP. It also moves the classifier to the Software defined networking (SDN) data plane, making use of features such as packet inter-arrival time.**

I. Introduction

Traffic monitoring is a necessary part of modern networks. With increased encryption and the dynamic nature of web protocols, this is not easy task. We propose a traffic classifier that will take different encrypted traffic and the traffic class, such as web browsing, VoIP, chat etc. Once we classify the traffic, we then generate flow rules based on that classification. Transferring the flow rules to a switch we are able to classify network traffic using the SDN data plane.

II. Background

Currently, the classification of network traffic is an important topic in the Computer Science domain as many network applications use it for monitoring network activity. By knowing what type of traffic is flowing in and out of a network, it can predict traffic patterns to provide more effective network capacities by categorizing it into 3 main categories: Sensitive Traffic, traffic that needs to be delivered and on time, Best-Effort Traffic, traffic that has a lower expectation for Quality of Service (i.e. packet loss), and Undesired Traffic, which is usually spam or malicious intent traffic.

With the increasing usage of network traffic such as Browsing, Email, Chat, Streaming, File Transfer, VoIP and TraP2P, classifying network traffic has now become critical. But, with the use of encrypting traffic, traditional network classifying techniques have now become ineffective for Internet Service Providers (ISP) and Network Administrators. Without the capability of predicting traffic patterns, network systems cannot provide optimal performances when distributing resources to different types of traffic, as they all have different Quality of Service Requirements (i.e. Bandwidth, Packet Loss, Jitter, and Delay) [1].

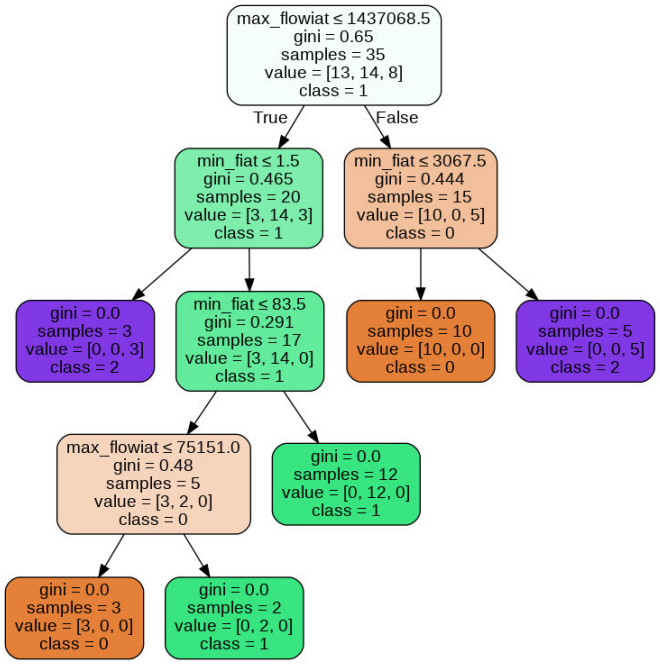

Fig. 1. Simple decision tree example with two features and 3 classes

III. System design

There are three control plane design areas to consider in our traffic classification model: data on which to train the ML model; training and testing the ML model; and converting the results of the ML model to information readable by the data plane. Below we outline these design areas as well as the design of the data plane itself.

A. Data Design

The dataset we used is from University of New Brunswick's Canadian Institute for Cybersecurity, which has VPN and non VPN datasets [2]. This dataset includes 14 traffic categories (VOIP, VPN-VOIP, P2P, VPN-P2P, Browsing, VPN-Browsing, Email, VPN-Email, Chat, VPN-Chat, Streaming, VPN-Streaming, File Transfer, and VPN-File Transfer). The traffic was captured from Wireshark and tcpdump, which used an external VPN service, that was then connected to OpenVPN. The original files are saved in an Attribute-Relation

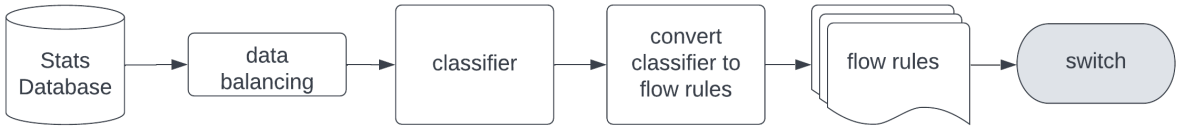

Fig. 2. Design of control plane from database to switch

File Format, which had 24 'columns' of attributes in each row. These attributes include: Duration, Total FIAT, Total BIAT, min FIAT, Min BIAT, Max FIAT, Max BIAT, Mean FIAT, Mean BIAT, FlowPacketsPerSecond, FlowBytesPerSecond, Min FlowIAT, Max FlowIAT, Mean FlowIAT, Standard FlowIAT, Min Active, Mean Active, Standard Active, Min Idle, Mean Idle, Max Idle, Standard Idle, and Classification. By using a Python script, we combined VPN and Non-VPN data, and parsed the 'columns', and translated them into separated Excel Sheets. A Jupyter Notebook File would grab the features we chose and put them into the ML Model. For our parsed data, we decided that an invalid value was a value of -1, meaning that we would not include any rows that contained a -1 in a column for our model. We decided on a main focus of Inter Arrival Time (IAT) for our features, as it is proven that IAT can increase classification accuracy substantially [3]. By using forward IAT (FIAT), backward IAT (BIAT), Flow IAT (FlowIAT), and separating them into their min and max values, we can create our thresholds for each classification. Additionally, doing a histogram of min/max features available, there were more valid values in Max FIAT, Max BIAT, Max FlowIAT, and Min FIAT, Min BIAT, Min FlowIAT, compared to Min Idle, Min Active and Max Idle, Max Active.

B. ML Model Design

For our machine learning model, we decided with ensemble supervised learning models, specifically decision tree (Gini) classifier. The reason of selecting Gini as a sub type was because entropy decision tree classifiers use logarithms, which makes the model performance slow [4]. We inserted our cleaned compiled data set for model creation, but came across an issue, the class distribution. In our dataset, the class distribution was highly imbalanced. We had 0.63% packets for Voice over Internet Protocol (VOIP) and 68.85% of browsing packets as shown in Fig.3

With such inequality, we had an issue with our model being overfitted with the browsing class variable. To resolve this, we used SMOTE library in python to equalize all the datasets. SMOTE oversampling technique created synthetic data points, which are duplicates of the class variable which are low in quantity and then adding them to the current dataset to remove imbalance. After applying SMOTE oversampling, our class distribution was equal as shown in below fig.4

Within decision tree classifier we started by initially selecting four features namely, Max FIAT, Max BIAT, Max FlowIAT and Max Idle and get accuracy results as shown below in figure 5, as you can see there are some unbalancing in

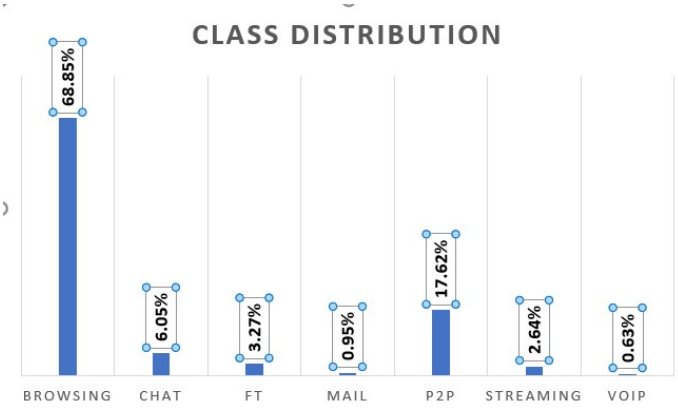

Fig. 3. Class Distribution before applying SMOTE

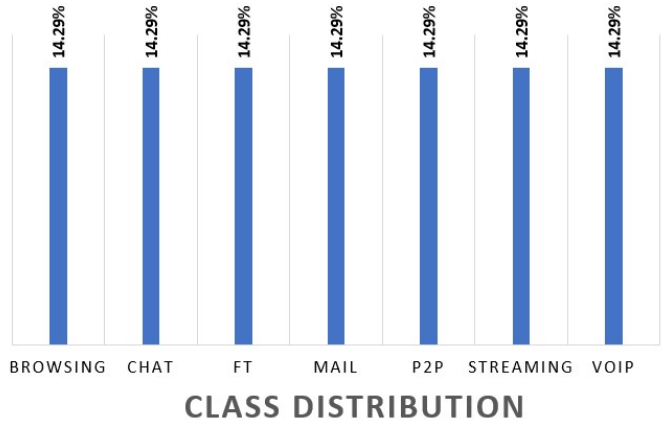

Fig. 4. Class Distribution after applying SMOTE

the accuracy results as data varies and accuracy is around 76.77%.

To improve our overall accuracy as discussed above (applying smote) and also increase the features to improve the accuracy, we updated our features to Max FIAT, Max BIAT, Max FlowIAT, Min FIAT, Min BIAT and Min FlowIAT. By increasing the features and applying the smote give us the accuracy result of around 94.5%. Our reasoning for choosing these features is firstly because for the initial iteration (4 features), we used the least amount of features that could give relatively good accuracy, and move forward from there. Secondly, there is a trade-off between accuracy, and switch resource utilization. The more features included, the more power a switch needs to process them. Thirdly, once we were

```
Accuracy :  76.77946324387398
Report :                precision    recall  f1-score   support

           0       0.88      0.86      0.87       599
           1       0.43      0.37      0.40        51
           2       0.41      0.70      0.52        20
           3       0.14      0.33      0.20         6
           4       0.66      0.61      0.63       156
           5       0.32      0.47      0.38        19
           6       0.17      0.17      0.17         6

    accuracy                           0.77       857
   macro avg       0.43      0.50      0.45       857
weighted avg       0.78      0.77      0.77       857
```

Fig. 5. Model Accuracy before applying SMOTE

able to reach a target accuracy (>90%) from the features we used, we did not update the features used for the ML model (a greedy approach). For future work, next steps would be introducing Feature Engineering to find and build more efficient models to gain the best accuracy with a manageable number of features.

```
  [  1   0   0 580   0   0   0]
  [  9   5   6   0 562   0   2]
  [ 10   6   5   0   0 574   1]
  [  3   4   2   0   1   0 576]]
Accuracy :  94.52821011673151
Report :                precision    recall  f1-score   support

           0       0.92      0.85      0.88       588
           1       0.93      0.91      0.92       601
           2       0.92      0.95      0.93       576
           3       0.99      1.00      1.00       581
           4       0.94      0.96      0.95       584
           5       0.95      0.96      0.96       596
           6       0.96      0.98      0.97       586

    accuracy                           0.95      4112
   macro avg       0.95      0.95      0.94      4112
weighted avg       0.95      0.95      0.94      4112
```

Fig. 6. Model Accuracy after applying SMOTE

C. Decision Tree to Flow Rules

The benefit of using a decision tree for classification is that the decisions made are comprehensible, and easily depicted using text or graphs. Fig.1 illustrates a simple example with two features, max_flowiat (maximum flow inter-arrival time) and min_fiat (minimum forward inter-arrival time). In this example there are 3 classes, 0, 1, and 2. To go left, the initial statement in the node must be true or to go right the statement must be false. For instance the root node of fig.1 will go left if a presented value is less than or equal to 1437068.5. At a leaf node there is an array of values representing each of the classes, with the highest array value being the most likely classification. Again, in fig.1 in the third row the left most leaf has a value array of 0, 0, and 3, indicating the most likely class is class 2.

To convert our decision tree to flow rules that can be used by the data plane we first divide the tree into left and right children. An array of leaf nodes is also constructed. These data structures are used to create an array of all possible branches within the tree. Each branch is traversed and the minimum and maximum value for each feature within the branch is stored in a feature min/max array. When a leaf node is reached the range between min and max, and the class decision for each feature is written to a text file as a flow rule. For instance a flow rule with three features and the class decision of 2 might look like *"table_add decision_table class_value 0->10 4->24 1->9 => 2"*. The rules.txt file and associated code is available in our repository on github.com [5]. As a way to facilitate collaboration between researchers during a pandemic, the control plane code, including the flow rules converter where developed using Jupyter notebooks and Google colab. The current version presented in our repository remains in that format.

As discussed in other areas of the paper we currently consider 6 features from the UNB VPN-nonVPN dataset [2]. These features may not be the best representatives of each class for our min-max branch approach and future work will be able to pin point the most efficient selections. The limitations of the p4 dataplane language must also be considered however. It is interesting to note that 6 features produces under 600 lengthy flow rules, while only relying on 1 feature produces over 6500 short rules. This is an important consideration when generating the rules, not just for the accuracy of our model but also when considering the memory limitations of SDN switches and the speed at which they are able to classify using our system. Another consideration is that the flow rules will need to be updated and pushed to network switches from the control plane as the nature of traffic changes over time. This correlation between number of features, number of rules, and model accuracy will be considered in future work.

D. Data Plane Design

The main task in data plane design is to pre-process the packets. The number and type of features of packets are key factor for classification, they would affect the granularity of decision tree directly. In this section, we describe steps in pre-processing and how pre-processing is important for classification. The only reason we pre-process packets is to extract the features. These features are necessary for the project. As shown in Figure 8, the steps in pre-processing are:

(1) Receiving the traffic
(2) Extract needed features by using register bank
(3) Use a match-action table to pass the features through flow rules and get the class value from it.
(4) Modify the customized header and attach it to the end of header field.

Since the basic logic of classification is that each type of traffic contains different features. We decide to use time-series statistics as the features for our project. Because time-series statistics are good when classifying protocols. In consider of learning curve and time cost, we give up using some powerful tools like NetMate or CICFlowMeter to extract statistical

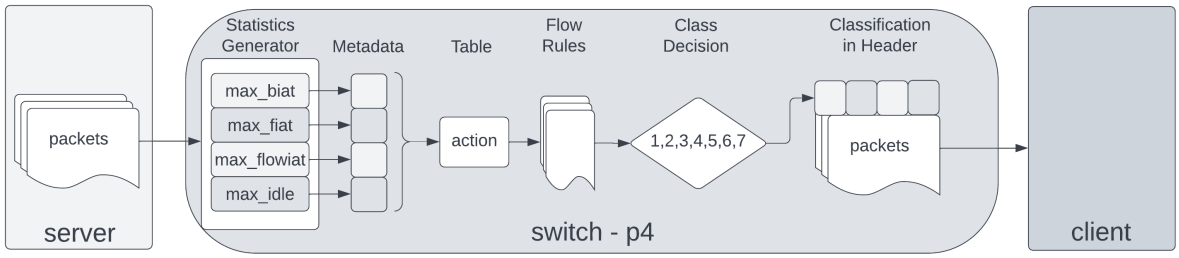

Fig. 7. Conceptual design of data plane from server to client

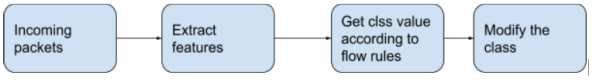

Fig. 8. Steps of pre-processing

features. But still, in the data plane, we generate 3 types of time-series statistics, 6 features in total. Forward Inter Arrival Time–FIAT (max and min) is the time between two forwarding packets. Backward Inter Arrival Time–BIAT (max and min) is the time between two backwarding packets. Flow Inter Arrival Time—FlowIAT (max and min) is the time between two packets regardless of direction. This means that traffic is bidirectional – incoming and outgoing, so we take the direction of flow as an important factor in pre-processing. We introduce a new mechanism to get the direction of packets. We use match-action table to implement the mechanism, it checks whether the source address or the destination address matches the local IP address, then it can invoke the corresponding actions to calculate the time-series statistics. The reason these features are important is that the granularity of decision tree will be better since we have more and more features. As a result, the better accuracy we can achieve.

IV. Evaluation

For evaluation purposes we implement our traffic classifier as a p4 application ($\approx$ 341 lines of code) within a BMv2 simple switch. Of key interest is the accuracy of our implementation. The source code is available at [5]. In this section we present our evaluation of the classifier prototype.

Evaluation was performed on a HP Z840 with 2 Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 2301 Mhz, 10 Core(s), 20 Logical Processors. On this hardware Oracle VirtualBox 6.1 was used to run the P4 Tutorial Release 2022-04-12 image. The operating system used was Ubuntu 20.04.4 LTS.

A. Setup

The VPN-noVPN dataset [2] provides data in two forms. The statistics provided by the dataset are used by us in creating the previously mentioned flow rules in Section III.C. We use the other data form provided in the dataset, pcap files, to evaluate our prototype. The pcap files are representative of the classes: *chat, file transfer, mail, P2P (peer to peer), streaming,* and *VOIP (voice over internet protocol)*. A seventh class, *browsing*, is not explicitly represented and as mentioned in [2] this class is an amalgam of the other classes. From the available pcap files we select 2 files from each class, excluding browsing. Within each class, 1 file is vpn traffic, and 1 file is non-vpn traffic.

With pcap files selected the files are reviewed in Wireshark Version 3.6.3 (v3.6.3-0-g6d348e4611e2) and the principle source and destination are identified. Using tcprewrite 4.4.1 these IPv4 addresses are changed to either 10.0.0.1 or 10.0.0.2 for testing convenience and consistency.

Our p4 dataplane file is compiled and the resulting simple switch started. The generated flow rules, and basic forwarding rules are then loaded onto the switch.

```
$ p4c --target bmv2 --arch v1model dataplane.p4

$ sudo simple_switch -i 0@veth0 -i 1@veth2 2@veth4 /
  --log-console --thrift-port 9090 dataplane.json

$ sudo simple_switch_CLI --thrift-port 9090 /
  /< ./rules.txt
```

To send and receive network traffic through the simple switch, Scapy 2.4.5 is used as seen in fig.9. The send file [5] cycles through an array of pcap files, sending packets into the simple switch on port 2 using virtual ethernet interface 4. Packets are received by Scapy receive files [5] listening on virtual ethernet interfaces 0 and 2. The receive files check the classification that was added to the packet by the simple switch and record it in a text file. The text files are used to compare the classification from the simple switch against the ground truth of the original pcap.

The accuracy of classification performed by our dataplane.p4 running on the simple switch is represented in tables I thru VIII. Each table shows a network traffic type and what the packets have been classified as.

Future work will involve comparing the accuracy and speed of our decision tree classifier running on a SDN switch to the same classifier running at the controller or a separate server. The implementation on a cpu of the statistical procedures used in [2] is necessary for this comparison and beyond the scope of the current research.

B. Results

Mentioned in the data modelling section, but worth noting again, are the results of our classification model. Using our

TABLE I
PACKET CLASSIFICATION FOR STREAMING (YOUTUBE) WITH 500 PACKETS RUN 10 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
stream_500	1	0	87	190	3	0	0	2	0	217	0	0	0	0	500/500
stream_500	0	0	84	193	4	0	0	1	0	217	0	1	0	0	500/500
stream_500	0	2	89	186	4	0	0	0	0	216	0	3	0	0	500/500
stream_500	0	0	88	191	2	0	0	0	0	218	0	1	0	0	500/500
stream_500	0	0	88	191	2	0	0	0	0	218	0	1	0	0	500/500
stream_500	0	0	164	83	6	0	0	0	0	195	0	7	0	0	500/455
stream_500	0	0	80	199	2	0	0	0	0	218	0	1	0	0	500/500
stream_500	0	2	82	192	5	0	0	1	0	217	0	1	0	0	500/500
stream_500	0	1	215	63	2	0	0	1	0	204	0	8	0	0	500/494
stream_500	0	0	212	67	2	0	0	1	0	213	0	5	0	0	500/500

TABLE II
PACKET CLASSIFICATION FOR STREAMING (YOUTUBE) WITH 1000 PACKETS RUN 10 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
stream_1000	1	7	328	155	3	0	0	2	0	392	0	10	0	0	1000/898
stream_1000	3	0	448	0	1	0	0	3	0	448	0	1	0	0	1000/904
stream_1000	6	34	354	120	2	0	0	1	0	404	0	14	0	0	1000/935
stream_1000	1	54	179	129	3	0	0	0	0	286	0	16	0	0	1000/668
stream_1000	0	0	149	392	7	0	0	0	0	451	0	1	0	0	1000/1000
stream_1000	0	2	306	237	3	0	0	0	0	439	0	13	0	0	1000/1000
stream_1000	0	15	207	167	4	0	0	1	0	322	0	6	0	0	1000/722
stream_1000	0	8	292	111	4	0	0	1	0	332	0	14	0	0	1000/762
stream_1000	0	1	196	349	2	0	0	2	0	447	0	3	0	0	1000/1000
stream_1000	0	2	354	100	7	0	0	2	0	378	0	13	0	0	1000/856

TABLE III
PACKET CLASSIFICATION FOR STREAMING (VIMEO) WITH 500 PACKETS (PAYLOAD REMOVED) RUN 10 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
stream2_500	1	0	203	0	8	0	0	3	0	113	0	8	0	0	500/336
stream2_500	0	0	190	143	2	1	0	1	0	163	0	0	0	0	500/500
stream2_500	0	0	213	122	0	1	0	1	0	163	0	0	0	0	500/500
stream2_500	0	5	139	47	5	1	0	2	0	82	0	7	0	0	500/293
stream2_500	0	0	193	142	0	1	0	1	0	163	0	0	0	0	500/500
stream2_500	0	0	141	187	7	1	0	2	0	162	0	0	0	0	500/500
stream2_500	0	0	205	128	2	1	0	1	0	163	0	0	0	0	500/500
stream2_500	0	0	225	33	4	1	0	2	0	107	0	18	0	0	500/392
stream2_500	0	1	256	72	3	1	0	1	0	145	0	14	0	0	500/493
stream2_500	0	0	201	131	3	1	0	1	0	163	0	0	0	0	500/500

original 4 features, we achieved over 75% accuracy, and with six features, we achieved over 90% accuracy. This speaks to a correlation between feature amount and accuracy, though we did not investigate this further since the value was high enough for our research. Provided that feature count effects accuracy, there is presumably a threshold where increasing features numbers is detrimental to accuracy and this is left to future work.

We also saw a successful result in translating the classification model to flow rules. A small inspection of the flow rules and a trace of several values resulted in correct classifications. This, however, is not a validation of the process and a more thorough validation needs to be carried out to determine if it is indeed accurate.

When testing began, it was found that the dataplane p4 file behaved differently when receiving single packets versus being fed a stream from a pcap file read by Scapy. Reasons for this were not readily identifiable, and the classification was also directed to the TCP:reserved header location to facilitate testing. Future work needs to identify the cause of this issue.

The first test of the complete system was done on a streaming data pcap, specifically YouTube, from the VPN-NonVPN dataset. We successfully altered client and server addresses to meet the requirements of our test bed. When the TCP packets with addresses 10.0.0.1 and 10.0.0.2 were run through the system, packets were assigned a classification by our system, but on closer inspection, it was found that there were multiple packet dropouts and the classifications were incorrect. To test whether packet dropout was related to system resources, we created a streaming data file with only 500 packets, and a file with 1000 packets using Wireshark. Each of these files was run through the test bed with the results as seen in Table I and Table II. The pcap with 500 packets had less dropout than the pcap with 1000 so it was decided that future

tests would only use 500 packets. The classifications remained incorrect in both files, with host 1 classifications clustering around file transfer (ft) and mail, with some outliers. Host 2, representing the server, mostly classified received packets as mail.

To test whether the incorrect classifications were isolated to the YouTube streaming file we ran a Vimeo capture through the system, and again the host 1 classification result clustered around ft and mail, with some noise in p2p. The host 2 classification was primarily ft, with some noise in p2p and several consistent packets classified as browsing. The number of packets sent versus the number received was fairly consistent, with only a few large dropouts. We ran this test again with the payload removed to see the effect. There was nominal change, as shown in Table III, which is to be expected since we are testing time related features.

The results of testing email, chat, ft, VOIP, and streaming were all similar, and similar to the original test discussed above. Classifications were incorrect. Host 1 classifications were clustered around ft and mail with a few outliers in p2p. Host 2 classifications were principally ft, with some outliers in p2p. These results can be seen in Tables III thru VIII. The sent to received ratio of packets was good, with a few exceptions where packet transmission dropped below 100. All classes were tested with 500 packets except for VOIP because of limited files of this type in the original dataset. Scapy returned errors that packets were too long in several of the classes when this set of tests was run so payloads were removed on all pcaps using pcap_remove_payload [6]

Table VI displays the results of the classification of file transfer (ft) packets. This run of tests is of interest because it is cleaner than all others, with classifications falling consistently into a single category, and the ratio of sent to receive packets being perfect. This type of traffic will be the best to use when trying to identify problems with our system in future work.

All the above tests were performed with non-VPN traffic. With VPN traffic, trying to rename sources and destinations using Wireshark was not possible. Tests were performed by manually changing the forwarding rules in our test bed. The limitation of classifications not being visible when pcap files were used, and the lack of a TCP:reserved area in VPN files meant no classification data was visible in transferred files. We have no results for VPN traffic.

The results from all our tests were positive in the sense that we received classifications. Results, however, were of little use since the classifications were consistently wrong. Packet dropout is a concern, as is classification clustering around only two classes.

V. Discussion

This section looks at the limitations of the research method used, specifically conclusion validity, internal validity, construct validity, and external validity. At the end we discuss future work.

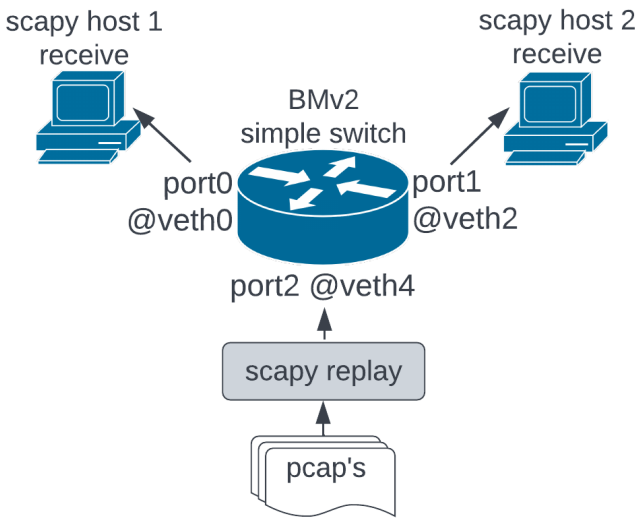

Fig. 9. Evaluation setup where pcap's are replayed through Scapy

A. Conclusion Validity

This research does not present a conclusion. It is not clear from the tests that network traffic can be classified using SDN switches. Some limited classification occurs in our tests but is incorrect and unclear if the reason for classification results from our data plane. No comparison system exists as a control against which to judge our results. Our decision tree run on a cpu is correct and validated by known tests. The number of runs of each data type within our test bed is arbitrarily chosen and the effect of more or less runs is unclear. The seemingly random dropout of packets is of concern.

B. Internal Validity

Where possible, uncontrolled external factors have been limited. Tests were run on a virtual machine with external network connections disabled and unnecessary services and programs ended. Still, the random dropout of packets demands a review of the test bed used. The amount of unvalidated outside code needed to prepare and launch the packets into the test bed is of concern. A BMv2 switch was used with default settings, which may not be the optimum for an unbiased result. Inexperience with the BMv2 switch may also result in incorrect usage. Data used for this research may contain bias, and may not be appropriate for the type of research being done. Statistics provided with the dataset were taken as is and not validated by us. Finally, because of pandemic restrictions, the researchers worked separately, with different sections of code being developed and then combined online and discussed in an online forum. Unforeseen problems may arise where code from different researchers crosses over and could be mitigated by more review and optimization of the code base.

C. Construct Validity

Using a decision tree for traffic classification is not new, nor is classification (of images) on SDN switches. The construct of

TABLE IV
PACKET CLASSIFICATION FOR EMAIL WITH 500 PACKETS (PAYLOAD REMOVED) RUN 5 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
email_500	0	0	77	188	6	0	0	2	0	226	0	1	0	0	500/500
email_500	0	0	12	20	5	0	0	0	0	34	0	2	0	0	500/73
email_500	0	0	70	195	6	0	0	2	0	226	0	1	0	0	500/500
email_500	0	0	91	179	1	0	0	0	0	228	0	1	0	0	500/500
email_500	0	0	94	171	6	0	0	0	0	228	0	1	0	0	500/500

TABLE V
PACKET CLASSIFICATION FOR CHAT WITH 500 PACKETS (PAYLOAD REMOVED) RUN 5 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
chat_500	0	17	72	72	0	3	0	0	0	0	175	0	3	0	500/342
chat_500	0	0	72	171	4	0	0	1	0	251	0	1	0	0	500/500
chat_500	0	0	22	26	6	0	0	0	0	37	0	4	0	0	500/95
chat_500	0	0	56	190	1	0	0	0	0	252	0	1	0	0	500/444
chat_500	0	0	64	176	7	0	0	1	0	251	0	1	0	0	500/500

TABLE VI
PACKET CLASSIFICATION FOR FILE TRANSFER WITH 500 PACKETS (PAYLOAD REMOVED) RUN 5 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
ft_500	0	0	0	162	5	0	0	0	0	332	0	1	0	0	500/500
ft_500	0	0	0	165	2	0	0	1	0	331	0	1	0	0	500/500
ft_500	0	0	0	165	2	0	0	0	0	332	0	1	0	0	500/500
ft_500	0	0	0	164	3	0	0	1	0	331	0	1	0	0	500/500
ft_500	0	0	0	163	4	0	0	0	0	332	0	1	0	0	500/500

TABLE VII
PACKET CLASSIFICATION FOR VOIP WITH 340 PACKETS (PAYLOAD REMOVED) RUN 5 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
VOIP_500	1	1	42	64	1	0	0	1	0	128	0	0	0	0	340/238
VOIP_500	0	0	99	10	4	0	0	1	0	121	0	5	0	0	340/240
VOIP_500	0	0	54	100	4	0	0	1	0	180	0	1	0	0	340/340
VOIP_500	0	0	47	107	4	0	0	0	0	181	0	1	0	0	340/340
VOIP_500	0	0	54	102	2	0	0	0	0	181	0	1	0	0	340/340

TABLE VIII
PACKET CLASSIFICATION FOR STREAMING WITH 500 PACKETS (PAYLOAD REMOVED) RUN 5 TIMES.

	host 1							host 2							
file	browse	chat	ft	mail	p2p	stream	voip	browse	chat	ft	mail	p2p	stream	voip	sent/received
Stream_500	0	17	19	95	2	0	0	0	0	265	0	6	0	0	500/404
Stream_500	0	0	0	164	3	0	0	0	0	332	0	1	0	0	500/500
Stream_500	0	20	5	23	5	0	0	0	0	104	0	7	0	0	500/164
Stream_500	0	1	46	104	4	0	0	1	0	305	0	3	0	0	500/464
Stream_500	0	4	77	36	6	0	0	1	0	236	0	5	0	0	500/365

representing the tree as flow rules by finding the minimum and maximum of each feature through traversal of tree branches is not fully validated, however. Further review of the correctness of this approach is needed. Also in question is classifying by time-based features, principally what features and how many are needed requires further research.

D. External Validity

The results are not generalizable outside the experimental setting. The BMv2, while good for initial testing, is not representative of real-world conditions. Data from the VPN, non-VPN dataset is good for testing but proved to be limited, and contains data for platforms, such as Vimeo, that are no longer representative of those platforms. Using only one switch, while convenient for testing, does not present the traffic collision and dropout problems that a live network would.

E. Future Work

To improve the accuracy of classification in the data plane, an increase in the diversity and number of features is necessary, for instance, idle time and active time. Also, some features could be measured within a given time, for example, bytes per second or packets per second. Instead of hard coding these features in P4, powerful tools like NetMate could be used to extract these features.

Of the four steps presented in this research, three need further validation. The existing dataset statistics and pcaps, conversion of decision tree to flow rules, and collection of time related statistics using a SDN switch are possible areas for introducing error. By proving the correctness of each of these, the research result will have more power. Random dropouts of packets during testing show that a review is necessary for the test bed as well.

This research has developed the data plane and an application that functions in the control plane. Developing how the application works with a controller, such as Ryu, is still necessary. Also needed is the design of how flow rules will be kept up to date by the controller, especially given the dynamic nature of network traffic.

VI. Related work

We did an ad hoc scoping review of recent work and found two themes relevant to our research: new approaches to traffic classification, and mapping ML to SDN. Research involving traffic classification is more mature than the mapping of ML to SDN, but we found interesting work in both themes that could be directly applied to our research.

The four papers we reviewed under the traffic classification theme took a research engineering approach with all performing a quantitative simulation [7][2][8][9]. Different motivations for traffic classification were identified. Traffic classification as essential to network management tasks like monitoring is found to be a motivator in Akbari et al.[7] and Gao et al.[8]. The motivator for traffic classification in Draper-Gil et al.[2] and Muehlstein et al.[9] is related specifically to security concerns. A key research problem in Akbari et al.[7] is the dynamic nature of web protocols and the increase in encryption shown by such protocols as HTTP/2 and QUIC. Proposed is a generic data-driven method that is protocol agnostic. Research in Gao et al.[8] cites the slow process of controllers getting statistics from the data plane as being unacceptable for tasks needing high reactivity when identifying traffic anomalies. To solve this, the researchers propose a data plane primitive that can efficiently track mean, variance, and standard deviation of traffic flow for use when identifying traffic based on its temporal characteristics. Muehlstein et al.[9] focuses on the security problem posed by a passive adversary eavesdropping on network traffic. Also concerned with security, [10] extend past work to show that applications supporting multiple services have added vulnerability to task classification.

Principally tackling the problem of increasing encryption, Akbari et al.[7] vectorizes flow statistics, shapes traffic based on size, timing, and direction, and the raw bytes from the TLS handshake packets of a real-world mobile traffic dataset from an ISP. They develop a hybrid neural network (convolutional neural network and a long short-term memory network) that uses extracted features to distinguish between the traffic classes of chat, download, games, and mail. They list the benefits of this approach as its small size compared to other ML approaches and its temporal nature. To facilitate in-switch statistical analysis, Gao et al.[8] create algorithms suitable for the computational abilities of SDN switches that can calculate mean, variance, and standard deviation. They then use these statistics to identify anomalies in the network traffic, specifically when one subnet is receiving more traffic than another. It is unclear which dataset this research used, but the algorithms have been combined into a publicly available library, which may be useful for our research. Support Vector Machine (SVM) is used in Muehlstein et al.[9] to classify traffic features identified from a dataset developed by the researchers. Of interest is the feature extraction process and the feature list developed for this research. Similar to Muehlstein et al.[9], Draper-Gil et al.[2] create a dataset, but with focuses on specific features of forward flows, backward flows, bi-directional flows, the idle-to-active and active-to-idle states, and number of packets per second. Two ML algorithms were used to gauge the success of the features for classifying, decision tree and k-nearest neighbors.

For the mapping of ML to SDN theme, we looked at two papers, Siddique et al.[11] and Xiong et al.[10]. Both papers tackle the ML to match-action pipeline problem, one presenting a general solution and one creating a specific computer vision (CV) related one. In Xiong et al.[10], four different machine learning algorithms, Decision Tree, SVM, Naive Bayes, and K-means, are evaluated on the v1model architecture using a BMv2 switch, Mininet, and a standard dataset. Siddique et al.[11] designed a complete image classification system using three components: ML classifier on a python based network controller, a P4 data plane using the v1model architecture on a BMv2 switch, and a client library that can encapsulate images into a readable packet format.

While the client library does not interest our research, the other two components contain potential starting points for our research when combined with the statistical analysis work of the other research mentioned. Of interest in the work of Siddique et al.[11] and Gao et al.[8] are the methods that overcome the lack of floating point operations available on software switches.

VII. Conclusion

This research presents a system of three designs that culminate in a classifier for encrypted network traffic using the SDN data plane. Our machine learning model design uses the ensemble supervised learning of a decision tree classifier. We refine a highly imbalanced dataset that was over-fitting using the SMOTE library, creating a well-balanced class distribution training set. By using four time related traffic flow statistics, our model achieves 76.77% accuracy and with six time related features, we achieve 94.5% accuracy. Our decision-tree-to-flow-rule design translates the decision tree model into flow rules useable on the SDN data plane. Our data plane design classifies live network traffic into one of seven categories based on time related features and lays the groundwork for finer grained classification.

Evaluation of our system reveals that errors exist in either or both our decision-tree-to-flow-rule design and data plane statistics generation. The overall system, however, successfully adds a classification to out-going packets, though the classification is incorrect, demonstrating the possibility of such a system.

As encryption and changing protocols make network traffic harder to classify, network administrators and other legitimate users who rely on classification to shape traffic will need tools such as our system. We have laid the groundwork for a complete system on the SDN data plane, as well as a test bed for refining that system, that with future research will perform an accurate classification of live encrypted network traffic without the use of specialized hardware.

References

[1] "quality of service requirements description," https://www.mdpi.com/1424-8220/21/14/4677, accessed: 2022-03-12.

[2] A. Habibi Lashkari, G. Draper Gil, M. Mamun, and A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," 02 2016.

[3] "Can we trust the inter-packet time for traffic classification?, howpublished = (http://www- sop.inria.fr/members/chadi.barakat/icc2011.pdf, note = Accessed: 2022-04-19."

[4] "Decision Trees: Gini vs Entropy, howpublished = https://quantdare.com/decision-trees-gini-vs-entropy/, note = Accessed: 2022-04-19."

[5] "encrypted-traffic-classification description," https://github.com/PINetDalhousie/encrypted-traffic-classification, accessed: 2022-03-12.

[6] "pcap-remove-payload description," https://gist.github.com/alem0lars/, accessed: 2022-03-12.

[7] I. Akbari, M. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "A look behind the curtain: Traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, pp. 1–26, 02 2021.

[8] S. Gao, M. Handley, and S. Vissicchio, "Stats 101 in p4: Towards in-switch anomaly detection," in *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, ser. HotNets '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 84–90. [Online]. Available: https://doi.org/10.1145/3484266.3487370

[9] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, and O. Pele, "Analyzing https encrypted traffic to identify user's operating system, browser and application," 01 2017, pp. 1–6.

[10] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, ser. HotNets '19. New York, NY, USA: ACM, 2019, p. 25–33.

[11] "Towards network-accelerated ml-based distributed computer vision systems," https://github.com/PINetDalhousie/netpixel.