Skype Covert Channel Identification using Sketches in SDNs

(extended abstract of the MSc dissertation)

André Madeira
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisors: Professor Luís Rodrigues and Professor Nuno Santos

***Abstract*—The characterization of network flows is relevant for multiple applications, in particular for security applications, such as the detection of covert channels in real time. Typically, this operation is performed by registering all the packets of the relevant flows and, later, analyzing their characteristics, for example, obtaining the distribution of their sizes. However, this solution consumes many resources, affecting network performance, and typically can only perform the classification at a later time. We evaluate the possibility of exploring the recent advances in SDN networks, programmable switches, the P4 programming language and probabilistic data structures (also called sketches) to characterize the flows in the switch itself thus reducing the amount of network data that need to be stored and analyzed to identify covert channels. We present a software architecture for programmable switches that allows us to characterize flows using two layers of filtering, each using a sketch, to first reduce the high amount of flows being processed, limiting the number of misclassified flows that do not contain covert channels. Our solution allows us to monitor 5K flows while keeping an accuracy of 0.95 in the detection of covert flows, representing an increase in analysis capacity of 20 times for the same amount of memory on the switch in the absence of a sketch.**

I. Introduction

The characterization of packet flows is required for various security applications, such as detection of covert channels [1], or creation of access profiles [2]. In this work, we address the problem of detecting, in real time, covert channels in multimedia flows, namely in Skype calls. We consider the scenario where it is possible to have access to packets exchanged for Skype connections and we intend to identify which of these are legitimate calls and which are calls that carry a covert channel. Covert channel information is typically encoded in audio and video exchanged by Skype through censorship-resistant communication tools such as Facet [3] or DeltaShaper [4]. As Skype encrypts multimedia information, the presence of the covert channel can only be detected by analyzing some features of the packets, such as their size or frequency. The thesis focus on the detection of covert channels based on the packet size distribution of the Skype call. Previous work [1] showed that most techniques used to create the covert channel are vulnerable to this type of analysis.

Typically, to analyze the distribution of the packets of a given Skype call, copies of the corresponding flow packets are created on a dedicated server that analyzes their characteristics. Since the number of Skype calls can be quite high, this process can entail a high cost, namely the bandwidth needed to collect copies of packets. In addition, the analysis of the packet size distribution is done in deferred mode, there existing significant latency in the detection of flows that carry covert channels.

In this work we assess the feasibility of exploring recent advances in computer networks to detect covert channels at lower costs and lower latency. SDNs provide the tools to change the behavior of switches at runtime, and therefore to dynamically select the flows for which statistical data is collected. We intend to take advantage of programmable switches, which can perform simple operations, to extract an approximate distribution of the packet sizes of the various flows in the switch itself without needing to duplicate packets. In addition, we use the P4 language to implement the programs capable of performing those operations at the switch. Since the amount of memory available in switches is relatively limited, we use probabilistic data structures (also called sketches) [5], [6]. These structures use the available memory in a very efficient way, but, on the other hand, they do not allow for characterization of the distribution of packets in a completely precise way. One of the challenges of this work is to understand how it is possible to increase the number of Skype connections monitored simultaneously, with the limited memory that exists in the switch, obtaining satisfactory accuracy in the classification of the flows.

Our solution is based on the use of a variant of the Count-Min sketch [6]. An experimental evaluation of our solution shows that it can significantly increase the number of Skype flows that can be monitored. Our solution allows the analysis of 20 times more flows for the same amount of available memory.

II. Related Work

This section introduces the concepts and technologies used in the development of our system, namely: software-defined networks (or SDNs), programmable switches, detection of covert channels and sketches.

A. SDN

The software-defined network architecture introduces a clear separation between what is called a data plane, responsible for transporting packets on the network, and the

control plane, responsible for deciding which packets should be transported and which paths they should follow. The data plane is realized by the network equipment, typically referred to as switches, which simply dispatch packets that are received at an entry port to one or more exit ports, through a dispatch table that takes advantage of specific fields of the packet such as the source IP, for example. Usually, these tables group packets into flows to identify them more easily. Sequences of packets transmitted between the same two computers and having the same fields used by the switch to differentiate them belong to the same flow. Dispatch tables can be configured remotely, via a standard interface, by a centralized controller. In this way, the routing policies can be expressed, programmatically, and executed by the controller. In practice, this controller is a program normally run on a dedicated machine. Switches can also collect some statistics on their operation that can be read using the same interface. This architecture has been proven to facilitate the configuration and monitoring of networks [7].

B. Programmable Switches

In a classic IP network architecture, the program run by a switch when dispatching a packet is set by the manufacturer and cannot be changed by the user. This restriction is due not only to commercial policies but above all to the need to ensure a high rate of packet switching. The evolution of technology has shown that it is possible to develop switches that can run small programs without compromising the speed of the system [8]. These switches allow the user to define programs to execute and allow these programs to be loaded dynamically. This makes it possible, for example, to install on the switch programs that perform flow monitoring functions tailored to the needs of the network operator.

C. P4

The Programming Protocol-Independent Packet Processors language, P4[9], is a high-level language for protocol-independent packet processing and data plane programming. P4 relies on the concept of match+action pipelines. Forwarding network packets can be broken down into a series of table lookups, until a suitable "match" is found, and modifications to protocol headers, which are known as "actions". P4 can change the way switches process packets once they are deployed, it is not tied to any predefined set of network protocols dependent of the switch hardware but any protocol of interest can be easily supported by the data plane programming, and programmers are able to describe packet processing functionality independently of the specifics of the underlying hardware. Additionally, through the P4Runtime API a centralized controller to communicate with the switch and modify entries in the match+action tables, update the forwarding plane logic and retrieve certain counter and flow information.

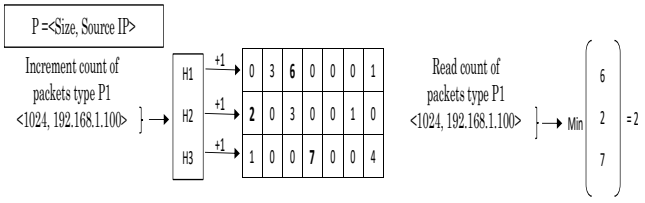


Figure 1. Representation of the Count-Min Sketch.

D. Covert Channel Detection in Multimedia Calls

A recent study [1] showed that the different techniques that are now known to establish covert channels in multimedia flows alter the packet size distribution, in relation to the distribution observed in a legitimate call. These differences can be detected automatically using supervised classification techniques based on decision trees, such as Random Forests or the *XGBoost* algorithm. This makes it possible to perform the detection of flows that cover channels by collecting the packet size distribution. If it is possible to capture this distribution, for example using the capabilities offered by programmable switches, it becomes possible to detect covert channels in real time.

E. Sketches

For large volumes of data, the calculation of exact values of traffic distributions is computationally expensive and requires large amounts of memory. The complexity of this task motivated the development of sketches, data structures that allow for the calculation of approximate values of metrics based on a large number of samples. Typically, sketches use hash functions to quickly associate the values read to registers, allowing the number of registers used to be significantly lower than would be necessary to obtain the exact value. However, the use of hash functions entails the possibility of collisions, in which different entities are associated with the same register, introducing an error in the estimate obtained. This error can be controlled by varying the number of registers and the number of hash functions used to realize the sketch. The literature is rich in different types of sketches [5], [6], and recently several of these structures have been proposed with the specific objective of facilitating the monitoring of computer networks [10].

F. Count-Min Sketch (CM)

The Count-Min Sketch [6] is a sketch designed to perform counts using memory efficiently. The sketch stores the counts in a two-dimensional matrix of registers, each row of the matrix being associated with a hash function. Figure 1 illustrates the operation of a CM sketch with three hash functions (H1, H2 and H3) while processing a packet identified by the tuple P = <*Size*, *Source IP*>. Upon receiving the packet P1 = <*1024*, *192.168.1.100*>, the sketch groups the values of P1 and uses each hash function to select the corresponding increments. Intuitively, knowing the space and amount of identifiers of the items to be recorded, it is possible to configure the number of rows and columns of

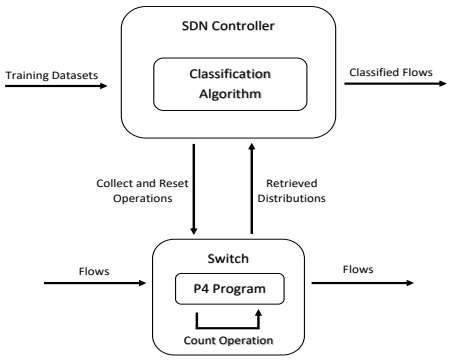
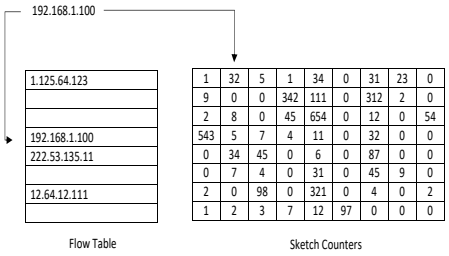


Figure 2. System architecture.



1	32	5	1	34	0	31	23	0
9	0	0	342	111	0	312	2	0
2	8	0	45	654	0	12	0	54
543	5	7	4	11	0	32	0	0
0	34	45	0	6	0	87	0	0
0	7	4	0	31	0	45	9	0
2	0	98	0	321	0	4	0	2
1	2	3	7	12	97	0	0	0

Figure 3. Flow Table Operation.

the matrix so that the probability of collisions in all rows is small (this value is a sketch configuration parameter). Finally, it is possible to read the P1 packet count recorded by the sketch by grouping the values belonging to the tuple and gathering the values of the registers corresponding to each hash function. The sketch returns an approximation of the actual count of P1 packets by choosing the lowest value obtained from each line. The reasoning for this decision focuses on the observation that the value of each line never returns a count lower than the actual value of the item and that higher values result from the occurrence of collisions.

III. Using Sketches to Capture Packet Size Distributions

This chapter introduces our architecture, a flow measurement system that allows for the monitoring of a high number of network flows while maintaining high accuracy for covert channel identification.

A. Goals

Our architecture works to facilitate the collection of flow metrics about packet length distributions to identify covert channels in Skype multimedia connections with high-precision as unobtrusively as possible and while keeping within the limitations of network devices. It makes use of sketches to reduce the amount of memory required while maintaining high accuracy for the metrics compressed therein. After obtaining the measurements at a network node (like a P4 switch), a separate machine should be responsible for performing the machine learning-based classification as seen in previous works [1].

B. Architecture Design

Our system, illustrated in Figure 2, makes use of programmable switches and SDN technology to create an execution pipeline capable of intercepting flows, counting packets, retrieving flow distributions and classifying them. The logic of our system is divided between two major components: the SDN controller and the P4 switch. Our system focuses on obtaining packet length distributions to populate the traffic differentiation mechanisms capable of detecting the existence of covert channels in multimedia flows as analyzed in Section IV.

The P4 switch is mainly responsible for the counting operation. It uses two different data structures: a Sketch whose counters measure the packet length distributions for the different flows. And a Flow Table to keep track of the identifiers of flows whose packets are being counted on the sketch counters. Each packet is identified through certain header field combinations and range of values their size fall into called buckets (for example, the packets of size between 10 and 20 bytes).

The SDN controller is responsible for periodically retrieving the metrics from the switch and resetting its data structures. The controller is programmed to execute a classification algorithm capable of identifying covert channels within Skype flows. In order to perform this operation, the algorithm must first be trained with datasets anticipating the conditions of the expected traffic, taking into consideration several factors. The execution of this algorithm leads to the classification of the analyzed flows with with varying accuracy according to these factors.

C. Switch Behaviour

The P4 switch is responsible for collecting the metrics necessary for the identification of covert channels and providing it to the SDN controller. The P4 program installed into the switch possesses two types of data structure, a flow table for storing the identifier of flows being measured and sketches which maintain the approximate distributions of the flows. The switch must be able to increment the sketch counters corresponding to each flow (updating the appropriate entry in the flow table if necessary), provide the flow representations to the switch and reset its data structures.

1) Flow Table: The flow table consists of an array of entries meant to store the flow identifiers of all the flows currently being measured. This identifier can be made up of different fields of the flow's packets headers like the TCP/IP 5-tuple.

The purpose of the flow table is to both maintain a list of all the flows being measured and to limit the number of concurrent flows being measured. The accuracy of the distributions is reduced the more flows are measured by the sketch. This can be due to having the same counters

being shared by more flows. Another reason for it can be due to reducing the amount of memory allocated to each flow sub-sketch as will see in the next section with one of our sketch variations. Since this list only maintains a set of identifiers its memory requirements are negligible when compared with the sketch's. As such we will not consider it in our calculations further on in the experimental evaluation.

Upon the arrival of a packet the flow table must first hash the identifier to find which entry the program should be occupied by the flow. If so, then the sketch will increment the corresponding counters. If it is not present, a new entry in the flow table needs to be allocated. If the entry is empty, the flow is added to the flow table and the sketch will increment the corresponding counters. Otherwise, the packet is ignored and forwarded as necessary. Without having to check each entry of the flow table one by one, the action of checking if the flow is already present in the table (and consequently, in the sketch) can be performed much faster. However, this will lead to collisions between flow identifiers which result in flows being ignored despite some entries still being available at the table.

Figure 3 shows the behaviour of our system when counting a packet identified by the IP 192.168.1.100 checking its corresponding entry in the flow table and finding that it is already present. In this case, the program will identify the respective counters and increment them. It should be mentioned that the above representation of the sketch counters is fully abstract and does not fully represent the real implementation of our sketches. Those will be explained in the following section.

2) CM Sketch Variations: In this section, we begin by describing an approach for the collection of flow metrics that does not take as a basis the use of a sketch and that therefore is expected to consume a large amount of switch memory. Then we introduced several variations of the CM sketch, describing the advantages and limitations of each variation for the accurate collection of metrics. The success achieved by each sketch in characterizing and differentiating different classes of flows is analyzed through the experimental study presented in Section IV.

Absence of Sketch (No-CM) In the absence of a sketch, the switch keeps a record for each packet size bucket belonging to a given flow. Each bucket corresponds to a range of values of sequential packet sizes. To accommodate the collection of a large number of flows, it is possible to represent the packet size distribution of each flow in a summary form by compressing this distribution by increasing the range of each bucket. However, it is possible that the use of this method may result in the inaccurate representation of the packet size distribution of each flow.

CM Sketch Simple (CM-S) The CM-S variation consists of a single CM sketch, the operation of which was previously described in Section II. When a packet is received by the switch, a register in the sketch is incremented corresponding to the application of the hash function on a tuple that includes: (a) the bucket value to which the packet size corresponds and (b) the flow identifier. We remind the reader that the identifier of a flow is composed of a set of fields common to all packets belonging to the flow, such as the source/destination IP and the ports used. In the CM-S sketch, the counting of packets of a given bucket belonging to a flow may be greatly influenced by packets (in this or another bucket) belonging to any other flow. Since each flow has a different identifier, the resulting tuple that is used as input to the hash function is always different, which can reduce collisions to values that still allow to differentiate the flows from the estimation of the distribution that results from the use of the sketch. Figure 4 illustrates the counting of a packet identified by tuple P1=<*1024*, *192.168.1.100*> in the CM-S sketch (on the left).

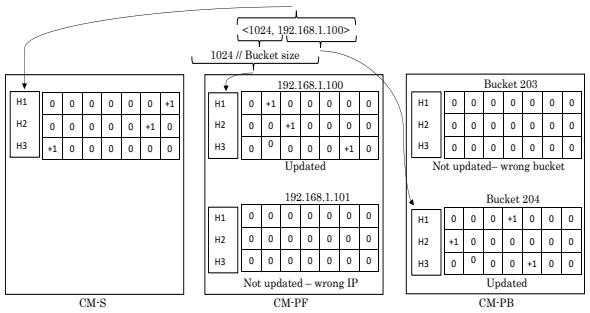


Figure 4. Example of the update of each sketch when receiving a packet.

CM Sketch per Flow (CM-PF) The CM-PF variation is composed of an CM sketch for each flow, each of which can be considered a sub-sketch of the CM-PF. When a packet is received by the switch, the switch checks the corresponding flow by attaching it to a specific entry in the sketch. This entry is determined by another hash function. If the entry obtained is already associated with another flow, the new flow is not registered. The register to be incremented is selected based on the application of the hash function to the bucket identifier corresponding to the packet size. Since this behaviour is similar to that of the flow table the same hash operation may be used to identify both the correct table entry and sub-sketch. The CM-PF sketch is based on the observation that collisions in the registers of each bucket will occur equally for each existing flow (since all CM sketches are configured with the same hash functions). Thus, by compressing a packet distribution into a limited set of buckets, it is expected that different classes of flows will cause a signature to emerge with sufficient information to differentiate the various classes. Figure 4 illustrates the counting of a packet in the CM-PF sketch (at the centre). Only the first entry of the sketch is updated since it corresponds to the flow to which the packet received belongs.

CM Sketch per Bucket (CM-PB) The CM-PB variation is composed of an CM sketch for each packet size bucket, each of which can be considered a sub-sketch of the CM-PB. Each sketch occupies the same memory. When a packet

is received by the switch, the CM-PB sketch increments the records corresponding to the application of the hash function on the flow identifier in the CM sketch for the size bucket of the packet concerned. In the CM-PB sketch, all the flows that collide in one of the sketches will collide in all sketches. However, if the colliding flows have similar distributions, these collisions do not affect the resulting distribution. On the contrary, if one of the colliding flows presents a distinct distribution, it may still be correctly identified, which is desirable in the context of this work. Figure 4 illustrates the counting of a packet received in the CM-PB sketch (on the right). Only the second entry of the sketch is updated since the bucket to which this sketch belongs corresponds to the size of the packet.

This concludes the discussion of the behaviour of the P4 switch and in the next section, we will begin discussing the behaviour of the SDN controller.

D. SDN Controller Behaviour

The SDN controller interacts with the switch by periodically retrieving the flow representations from the switch and resetting its data structures. Both of these operations are supported by the P4Runtime API. Afterwards, the metrics are classified by a classification algorithm, which is trained with a profile appropriate for the gathered flows. Next, we provide more details on how to generate training datasets and how the classification algorithm works.

1) Configuration Space and Dataset Generation: In order to train the classifier to correctly identify the we must generate training datasets based on the configuration employed. This configuration is meant to emulate the certain properties of the execution environment our system is to be employed in. This configuration is based on several parameters:

- *Sketch Variation*: Different sketch variations cause a different impact to the accuracy of flow identification since they allow the different flows to collide in different ways. Our datasets are retrieved from the results of applying the sketch variation compression to a non-compressed set of flows.
- *Available Memory*: The memory available to the sketch is dictates the total number of counters available to the sketch, therefore directly impacting the final accuracy.
- *Sketch Size*: Altering the width and length of the sketch (or sub-sketches) will alter the error margins according to the CM sketch [6] specifications.
- *Maximum Flow Number*: The number of concurrent flows can severely impact the accuracy of the classification since a too great number of different flows colliding on the same counters can lead to misleading factors towards the classification. For the CM-PF this will lead to a greater number of sub-sketches thus reducing their size and possibly compromising the distributions of their flows. The maximum number of flows can be enforced by the number of entries in the flow table.
- *Regular-Covert Flow Ratio*: Although this is of no significance to the CM-PF variation, both the CM-S and CM-PB variations can be severely impacted by this ratio since both covert flows and regular flows may collide with each other in these variations. In the cases where multiple flows of a single type collide with very few of another type, the latter may be incorrectly identified as the former. This is due to the more common type of flow being a greater influence on the final mixed result leading to it resembling those flows.
- *Packets per Flow*: This can also be interpreted as the amount of time between each retrieval of metrics. The more packets per flow we take in the more descriptive the packet length distribution becomes. However, similar to the previous item, in cases of multi-type collisions it may lead to misclassifications.
- *Bucket Range*: The range of values of each bucket impacts all sketch variations. Higher bucket ranges will lead to less overall buckets to maintain while providing less detailed distributions and vice-versa. For the CM-PB, a greater number of buckets will lead to a greater number of sub-sketches thus reducing their size and possibly compromising the distributions obtained.

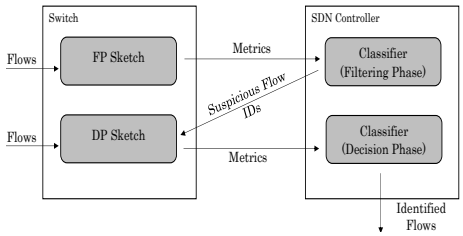


Figure 5. System architecture.

Datasets with the the packet size distributions that take into account these parameters should be generated and used to train the classifier before the system begins its execution. Additionally, generating testing datasets using the same parameter configuration can be used to test the accuracy of the classifier until acceptable results can be achieved. This way the classification algorithm may begin accurately identifying covert channels as soon as they are retrieved from the switch.

2) Classification Algorithm: The classification algorithm allow us to sort the received flows into the two classes: Regular skype flows and Covert channel flows. It does so by taking into consideration the different statistics of each type of flow across several instances present in the training datasets. In our case these statistics consist of the several buckets that make up the packet length distributions. Of course, this means that altering the number of buckets will provide us with with varying numbers of datapoints. By reducing the number of datapoints we can greatly reduce the error introduced by the sketches since the number of colliding items is reduced as well.

E. 2-Phase Architecture

In addition to the previous architectural design, we propose an extension to it in the shape of a two-phase architecture, both working in parallel, as seen in Figure 5. It uses two flow tables, two sketch variations and two classifier configurations in sequence in order to identify flows carrying covert channels. The first phase is called the *Filter Phase* and the sketch belonging to it is called *FP Sketch*. The second phase is called the *Decision Phase* and the sketch belonging to it is called the *DP Sketch*. Later, in the evaluation section, we will assess the effectiveness of this architecture against that of the single-phase architecture and what type of sketch is best suited for each phase.

In the two-phase architecture, certain suspicious flows can be processed by both phases. New flows are first inserted into the flow table associated with the FP Sketch and start incrementing counters in the sketch itself depending on the functioning of the variation. The purpose of this phase is to obtain a preliminary (and possibly more inaccurate than when using the single-phase architecture) estimate of the packet length distribution of each flow. Additionally, similarly to the regular architecture, the content of this sketch will be periodically retrieved by the controller in order to be classified. However, after the classification algorithm identifies the class of each flow, the regular Skype flows are discarded while the ones identified as containing a covert channel are kept. These flows are considered suspect and will be inserted into the flow table of the DP Sketch. This is performed during what was previously the "reset" operation and while the flow table of the FP Sketch will be completely reset, the one from the DP Sketch simply has its contents overwritten. Both sketch variations are still reset.

For the Decision Phase, the flow table associated with its sketch does not function the same way as its original incarnation. Instead, upon the reception of a packet belonging to a flow present in the flow table, the DP will increment its corresponding counters in its sketch. If the flow is not present in this flow table it may still be inserted in or matched to the flow table corresponding to the FP Sketch and increment the counters in the FP Sketch according to its original behaviour.

The purpose of the DP Sketch is to obtain a second estimation (usually more accurate than the previous phases' since it only has to account for the previously classified suspicious flows) of the reintroduced flows, thus providing the system with two layers of filtering out flows that the classifiers does not consider suspicious.

Seeing how regular Skype flows are more common than ones containing covert channels, the Filtering Phase contributes to softening the initially biased ratio between the two before performing a precise classification with the Decision Phase. Hence since the number of flows measured in the Decision Phase is much smaller than in the Filtering phase we can obtain much more precise metrics for the suspicious flows which can be used to better identify covert channels.

Finally, since both phases will have to handle significantly different number of flows the memory available will not be equally divided between the two sketches. Instead, a larger part of the available memory is allocated towards the FP Sketch and the remainder to the DP Sketch. Additionally, due to this separation of the available memory it is also likely that the classifiers for each phase use different configurations when creating the training datasets.

IV. Evaluation

The evaluation of our system. It highlights our architectures ability to perform flow distinction between those that contain covert channels and those that do not. In addition it highlights the ability of an architecture that contains two filtering phases to further reduce the number of misclassified flows that do not contain covert channels. Our evaluation is mainly concerned with two dimensions: the number of flows that can be characterized and the accuracy of the differentiation associated with these flows.

A. Experiment Configuration

1) Sketch Configuration: We simulate the use of a programmable switch using sketches for the collection of a representation of packet size distribution for a limited number of flows. Each sketch uses three hash functions that correspond to the application of a single hash function initialized with a distinct seed value.

The useful memory of the switch will be set to the total size of 0.3 MB, which was chosen chosen taking into account the theoretical basis underlying the design of the CM sketch. Measuring 1000 flows of approximately 3000 packets each certain using 0.3 MB of memory provide us with certain guarantees. With this amount of memory it is guaranteed with 95% probability that the value of the count read from each register, in a given bucket of the CM sketch, incurs an error of less than 10% of the total package of the true distribution of a flow.

To match the configuration in previous work [1], the packet sizes of each flow are quantified in buckets at the granularity $k = 5$ bytes. Considering the possible existence of packets with a size ranging from 0 to 1500 bytes, we use a total of 300 buckets. Additionally, we use the source IP (randomly assigned) to assign an identifier to each flow.

2) Dataset: Our experiences contemplate the transmission of multimedia flows in a proportion of 95% of legitimate Skype flows and, respectively, 5% of Skype flows carrying a covert channel. This ratio captures the simulation of a real workload where it is expected that the vast majority of the observed flows in the network do not act as vehicle of a covert channel. The flows that carry a covert channel were produced by the Facet [3] and DeltaShaper [4] tools, which replace a region of the video frames produced in legitimate video calls with some content to be transmitted in the covert channel, e.g., a Youtube video or a stream of IP packets. Each flow has a total duration of 60 seconds. All flows were obtained through contact with the authors of the study mentioned above [1] and number at 1000 Facet flows and 300 flows for two configurations of DeltaShaper. Since all

of the available flows have an approximate duration of one minute we split them into segments of 30 seconds, one for each phase of the 2-phase architecture. Using the same 30 second segment during the single phase architecture allow us to compare the two architectures.

3) Metrics: Previous works [1], [4], [3] studying the detection of covert channels on the Internet use the ROC AUC [11] metric to measure the success achieved in the differentiation of flows. Briefly, AUC summarizes the relationship between a classifier's true positive and false positive rates. In our case, the true positives would be the amount of Skype flows identified as such and the false positives would be the same for the covert channel flows. In the context of this work, a classifier with the ability to make a random guess about the class of a flow displays an AUC = 0.5, while a perfect classifier is characterized by an AUC = 1. We use the same metric to evaluate the accuracy of the different types of sketches. Additionally, we use the supervised classification algorithm *XGBoost*, as proposed by Barradas et al. [1]. The classifier is trained using a set of flows reserved for training, collected in rounds. In each round, we select a flow sub-sampling at 95%/5%. Then an approximate representation of the distribution of these flows is obtained through the sketch under analysis. The classifier is trained with a set of balanced samples (with the same amount of Skype and covert channel flows) obtained from the representations of the distributions generated by the sketch. This process is repeated for 200 rounds.

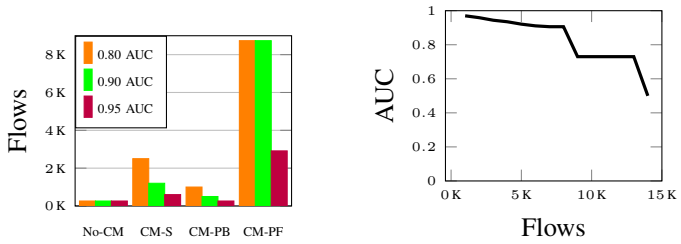
B. Single Sketch Evaluation

1) Setup: We begin by comparing the performance of the different variants of the CM sketch to a solution that does not resort to sketches. All variations use all available memory. In this case, the first 30s of traffic corresponding to each flow were collected by the sketch with 0.3MB of memory available. In this experiment, we intend to understand the number of flows that can be analyzed by the different types of sketches while obtaining a classification accuracy above three different values of AUC appropriate for each system.

2) Results: The results are shown in Figures 6, 7, and 8. In the graphs on the left we compare all the different sketches with each other while in the graphs on the right we track the AUC values for the best performing sketch variation as we increase the amount of flows being tracked.

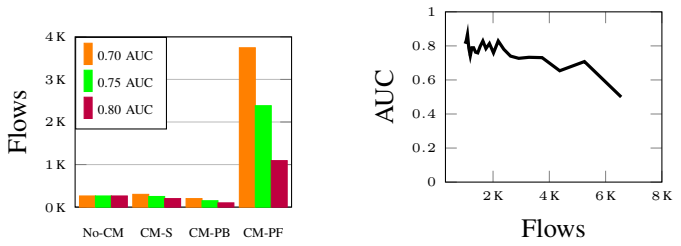
All systems show somewhat similar results in Figures 6a, 7a and 8a. In all three, CM-PF provides the best performance, showing significantly better results than the other variants, CM-S appears to be the next best variation closely followed by CM-PB, both of which often show results similar to No-CM in all figures.

Facet, as seen when compared with the other two systems, shows the best results, being able to support close to 3K flows concurrently at the highest AUC value we set for it (0.95). We believe this is due to Facet possessing the highest base accuracy (where the metrics are collected without compression) at approximately 0.99 AUC whereas DeltaShaper 320 possessed a base 0.86 AUC and DeltaShaper 160



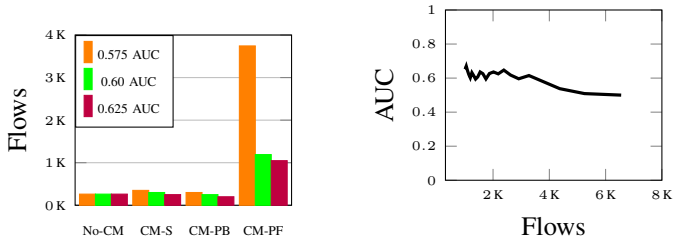
(a) Flows (b) ACU

Figure 6. Performance of CM sketch for Facet.



(a) Flows (b) ACU

Figure 7. Performance of the CM sketch for DeltaShaper 320.



(a) Flows (b) ACU

Figure 8. Performance of the CM sketch for DeltaShaper 160.

possessed 0.66 AUC. Since employing sketches incurs some form of compression on the metrics the poorer the initial accuracy of the system, the poorer the results will be after they are compressed. In the absence of sketches, with the previously defined configuration, it is possible to measure a total of 262 concurrent flows. Since no sketch is used the AUC value is always equal to the base AUC of each system.

The CM-S sketch can measure more flows than simple registers for all AUC values, indicating that despite the various collisions between flows and buckets it is still possible to distinguish the two types of traffic with high accuracy. The CM-S sketch was able to successfully measure approximately 500 flows concurrently for the Facet system at highest AUC value, while for the other two it obtained results equal to that of the No-CM variation.

The CM-PB sketch shows values below CM-S, which is due to the fact that CM-PB does not make efficient use of the available memory, since the different buckets are not used uniformly by the different flows. This phenomenon can be seen in Figure 9, which illustrates the heat map for buckets 13 and 100. Each line corresponds to the application of a

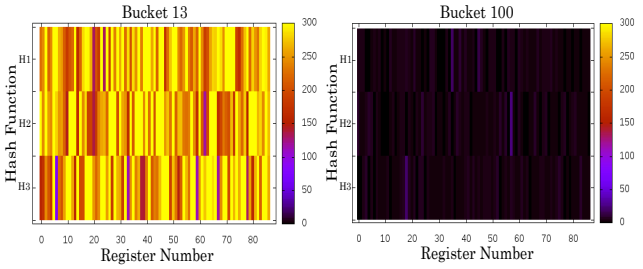


Figure 9. Heat Map for buckets 13 and 100 of the CM-PB sketch.

distinct hash function (as previously illustrated in Figure 4) and the memory division provides 87 registers for each hash function for each bucket. According to the heat map, this number of records is not sufficient to avoid a large number of collisions as the number of flows accounted for by the sketch is increased. It is then found that for the flows under analysis there are buckets with a high packet count while others have a low count. The redistribution of the space reserved for buckets rarely used to reduce the number of collisions in the remaining buckets is a time-consuming process that requires manual tuning of the sketch. The CM-PB sketch was able to successfully measure approximately 200 flows concurrently for the Facet system at highest AUC value, while for the other two it obtained results equal to that of the No-CM variation.

Finally, the CM-PF sketch gets the best results. This is due to the fact that each flow is kept separate from all others, preventing the collision between legitimate Skype flows and flows carrying a covert channel. Additionally, the two types of flows are easily distinguishable based on the packet size distribution, and these differences remain visible even when several buckets collide in the same register. It should be noted that CM-PF supports the same number of flows for AUC 0.8 and 0.9 as the compression at the register's level is not significant. The CM-PF sketch was able to successfully measure approximately 3K flows concurrently for the Facet system at highest AUC value, while for the other two it was able to measure approximately 1K flows.

For the graphs 6b, 7b and 8b we can see how as the packet length distributions are further compressed when using the CM-PF sketch, the accuracy decreases accordingly. By using CM-PF to measure large quantities of flows it is necessary to reduce the size of each CM sketch, leading to that, from a certain number of flows, the number of collisions between distinct buckets leads to loss of ability to distinguish traffic.

The results shown in Figure 6b show that the CM-PF sketch reaches an AUC of between 0.9 and 1 for a number of flows of less than 8K, falling to 0.7 at 13K flows. At this point, a sharp decrease in AUC is observed, approaching the value 0.5, and therefore reducing the classification decision to a random guess for a number of flows greater than 14K. For both the DeltaShaper graphs, the expected decrease is not as linear as for the Facet system. In accordance with our previous observation, of inferior base AUC's leading to decreased accuracy when performing compression, so too does the progress of the AUC become more unpredictable as we add more flows. In short, the figure shows that as the size of each CM sketch decreases, each counter that composes it will be shared by a greater number of buckets. In this way, the values corresponding to reading these buckets will tend to be shared. Of course, the more buckets share the same values, the greater the degree of overlap of approximations of distributions of different flows. When the system in question possesses a high base AUC, the decrease in the AUC as more flows are added seems almost linear. Something which becomes less possible as the base AUC decreases.

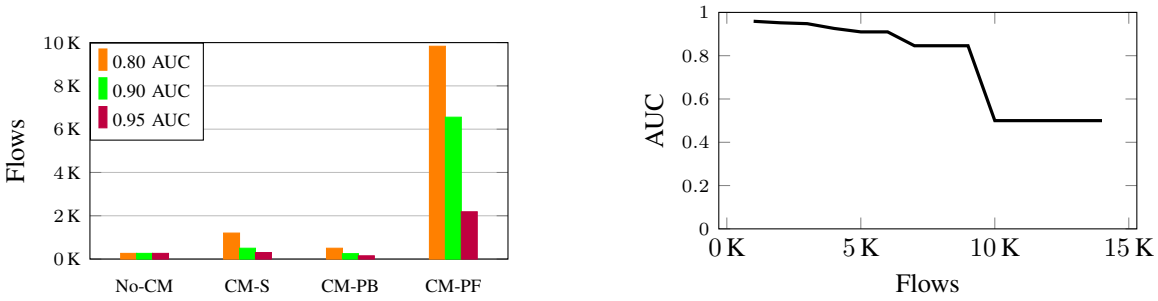
C. Hash Function Variation

1) Setup: Next, we seek to investigate how the addition of an extra hash function impacts the accuracy of the classifier while maintaining all other parameters of the previous configuration. We utilize the Facet test bed as it showcased the best results in the previous tests and possesses the highest base accuracy of all three systems. By increasing the number of hash functions we increase the likelihood of the query being within a certain error margin from approximately 95% to 98% while increasing our error by approximately 25%.

2) Results: While the results from the graphs in Figure 10 seem similar to the previous a closer inspection shows that the addition of a fourth hash function was unable to produce superior results to the previous test's, in some cases there was a decrease of almost 50% of flows measured concurrently. Increasing the number of hash functions results in a decrease in the number of counters per sketch row which results in more collisions. This increase in the number of collisions leads to an increase in the expected error margin for each item read by approximately 25%, resulting in metrics further compressed than seen in previous tests.

From graph 10a, when compared with 6a, we can observe a decrease in the number of flows that can be concurrently measured for both the CM-S and CM-PB sketches. For the CM-PF sketch the same can be seen for the AUC values of 0.90 and 0.95. While 0.80 AUC does show an increase, this is due to it being an extreme case where neither sketch could be pushed anymore allowing the 4 hash functions case to surpass the 3 hash functions case due only to having just enough counters to provide a decent accuracy score. In this case, adding any further compression would have left the system with the probability of correctly identifying the type of flow being classified is similar to that of a coin flip, therefore what we see at 0.80 AUC is an outlier that does not fully represent the impact of the additional hash function. Graph 10b also shows a much more rapid drop in accuracy than with only 3-hash functions and an overall decrease in the amount of measured flows.

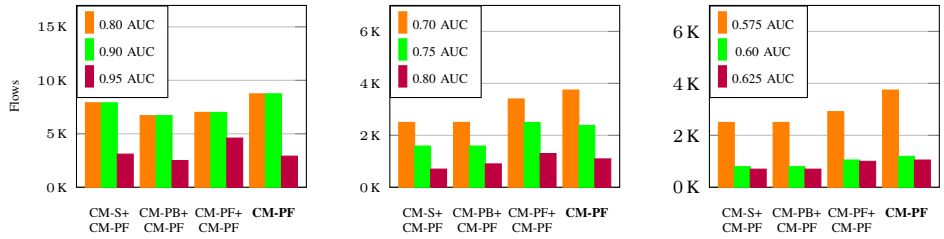
Overall, it seems that for our purposes increasing the number of hash functions does not provide us with any benefits. The increase of the likelihood of the query being within a certain error margin does not compensate for the added error due to having less counters per hash function.



(a) Comparing all sketches.

(b) CM-PF progression.

Figure 10. Analysis capacity of flows for each variation of the CM sketch for the Facet system using 4 hash functions.



(a) Facet

(b) DeltaShaper 320

(c) DeltaShaper 160

Figure 11. Analysis capacity of flows using the two-phase architecture.

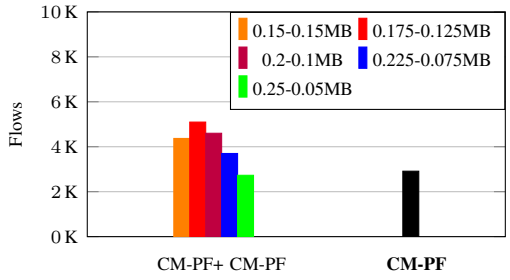


Figure 12. Analysis capacity of flows using the two-phase architecture.

D. Evaluation of the Two-Phase Architecture

1) Setup: In this section we assess whether the proposed architecture, which is based on the use of two sketches in sequence, offers advantages in relation to the use of a single sketch. Instead of experimenting with all possible combinations of different sketches, we chose to always use the sketch that presented the best results in the decision phase (i.e. using CM-PF as an DP sketch) and to vary only the sketch used in the filtering phase. To simulate the differentiation of flows in real time, the filtering phase is fed with the first 30s of traffic of each stream, while the decision phase is fed with the remaining 30s (making up the total duration of each flow in our data set).

2) Results: In Figure 11 we present the results for a configuration where 0.2 MB are reserved for the FP sketch and 0.1 MB are used for the DP sketch for all 3 systems. Note that in these tests only the DP sketch maintains the indicated AUC; the first phase may admit a lower AUC as the flows selected for the second phase will be re-analyzed.

Taking as an example the 9K flows Facet can collect for AUC values of 0.80 and 0.90 using 0.3MB as seen in Figure 6a, the two-phase architecture should, therefore, be able to classify 3K flows when using 0.1 MB in the decision phase. In this way, to be competitive, the architecture in two phases will have to filter more than 2/3 of the flows in the filtering phase. The values of Figure 11 show that this is not always accomplished. Specifically, for lower AUCs, like 0.80 and 0.90, the two-phase architecture cannot classify the same number of flows as seen previously using a single CM-PF sketch.

On the other hand, when aiming for 0.95 AUC values, the two-layer architecture can offer tangible advantages. In fact, using a combination of the two-layer CM-PF sketch it is possible to increase the number of flows that can be monitored from 3K to about 4.5 K, which represents an increase of about 50% for the Facet system. Seeing as the original 3K flows for this AUC value is much lower than the other values, the 4.5K flows seen here is a sufficiently low amount that the filtering phase can successfully filter out a great majority of the flows.

The only tools that show increases over the single phase architecture are the Facet and DeltaShaper 320. This relates once more to the base accuracy of each system. Since the accuracy of DeltaShaper 320 and 160 are inferior to Facet's, the number of flows that are filtered out by the filtering phase are lower than Facet's as well. This means that a higher percentage of flows are considered suspicious and pass to

the decision phase. Since the number of flows that can be measured concurrently in the DP sketch is limited, the initial number of flows fed to the filtering phase must also be lower in order to account for this limit. As such, DeltaShaper 320 (which has a base AUC of 0.86, lower than Facet's 0.99) does not show an increase in flows of the same proportion as Facet's. Additionally, DeltaShaper 160 (with a base AUC of 0.66) does not show any improvements at all.

E. Varying the Filter's Memory

1) Setup: In this section, we analyze how the available memory for the different phases of analysis affects the ability that the system displays in the differentiation of flows. More concretely, we compare different configurations of the architecture in two phases, varying the proportion of memory reserved for each of the sketches, namely by fixing the available memory for FP, between 0.15 MB and 0.25 MB, in successive increases of 25KB. In this experience, we use the best combination of previously identified sketches (CM-PF + CM-PF) and best performing system (Facet).

2) Results: As can be seen from Figure 12, different proportions show different results, as the memory reserved for the DP needs to be sufficient to accurately classify all flows that are not filtered in FP. Therefore there is a balance between the filtering capacity of the first stage and the accuracy of the second stage. For the study settings, the proportion that offers the best results is to reserve 0.175 MB for FP and 0.125 MB for DP. With this configuration it is possible to monitor about 5k flows, a gain of about 66% relative to the use of a single sketch.

V. Conclusions

In this work we studied the possibility of classifying covert channels in real time and efficiently, taking advantage of the use of programmable switches, software-defined networks and the P4 language to capture an approximate representation of the distribution of the flows to be monitored. This representation is then sent to a server to be classified. In order to capture an approximation of the distribution efficiently, we use probabilistic data structures known as sketches. In this context, we propose an innovative two-layer filter architecture, where in a first phase a sketch allows for filtering a significant fraction of the flows, the remaining flows being classified using a sketch configured to obtain better accuracy. At its best, this architecture shows gains of more than 66% from the use of a single sketch, being able to monitor about 5k flows simultaneously and offering an AUC of 0.95 in the identification of covert channels, using for this purpose only 0.3 MB of memory in the switch.

Acknowledgments

This work was partially supported through projects with ref. UID/CEC/50021/2019 and COSMOS (financed by the OE with ref. PTDC/EEI-COM/29271/2017 and by the Programa Operacional Regional de Lisboa in its FEDER component with ref. Lisboa-01-0145-FEDER-029271) and by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. Parts of this work have been performed in collaboration with Diogo Barradas and other members of the Distributed Systems Group at INESC-ID.

References

[1] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," *Proceedings of the 27th USENIX Security Symposium*, pp. 169–185, August 2018.

[2] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium*, Austin, Texas, USA, August 2016, pp. 1187–1203.

[3] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, Scottsdale, AZ, USA, 2014, pp. 163–172.

[4] D. Barradas, N. Santos, and L. Rodrigues, "Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams," in *Proceedings on Privacy Enhancing Technologies*, vol. 2017(4), Minneapolis, MN, USA, 2017, pp. 5–22.

[5] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *ICALP '02 Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pp. 693–703, July 2002.

[6] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, April 2005.

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, A. S., and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[8] Barefoot, https://www.barefootnetworks.com/products/brief-tofino/, accessed: 2019-10-26.

[9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, July 2014.

[10] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," *SIGCOMM '18 Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 561–575, August 2018.

[11] T. Fawcett, "Roc graphs: Notes and practical considerations for data mining researchers," *ReCALL*, vol. 31, pp. 1–38, 01 2004.