Envisioning a Unified Programmable Dataplane to Monitor Slow Attacks

Cuidi Wei
George Washington University

Shaoyu Tu
University of California Riverside

Toru Hasegawa
Shimane University

Yuki Koizumi
Osaka University

K. K. Ramakrishnan
University of California Riverside

Junji Takemasa
Osaka University

Timothy Wood
George Washington University

***Abstract*—Recent work shows that programmable switches can effectively detect attack traffic, such as denial-of-service attacks in the midst of high-volume network traffic. However, these techniques primarily rely on sampling or sketch-based data structures, which can only be used to approximate the characteristics of dominant flows in the network. As a result, such techniques are unable to effectively detect low-volume attacks that stealthily add only a few packets to the network.**

Our work explores how the combination of programmable switches, Smart network interface cards, and hosts can enable fine-grained analysis of every flow in a network, even those with only a small number of packets. We focus on analyzing packets at the start of each flow, as those packets often can help indicate whether a flow is benign or suspicious. We propose a unified architecture that spans the full programmable dataplane to take advantage of the strengths of each type of device. We are developing new filter data structures to efficiently track flows on the switch, dataplane-based communication protocols to quickly coordinate between devices, and caching approaches on the SmartNIC that help minimize the traffic load reaching the host. Our preliminary prototype can handle the full pipe bandwidth of 1.4 Tbps of traffic entering the Tofino switch, forward only 20 Gbps to the SmartNIC, and minimize the traffic load to 5 Gbps reaching the host due to our efficient flow filter, packet batching, and SmartNIC-based cache.

***Index Terms*—Traffic monitor, slow network attacks, programmable switches, smartNIC**

I. Introduction

New technologies such as P4 switches, SmartNICs, and virtualized middle boxes are adding tremendous power to the network data plane. However, appropriately harnessing these heterogeneous processing capabilities remains a major challenge, as their capabilities vary widely. Only by accounting for the strengths and weaknesses of each of these platforms, will we be able to find the proverbial 'needle-in-the-haystack' when monitoring large traffic volumes.

Programmable switches are increasingly being suggested for monitoring, using queries processed at Terabit link rates [1], yet their small memory (10s of megabytes) limits their ability to perform stateful processing. SmartNICs allow packet processing at fairly high-speed, i.e., 40–100 Gbps [2], although not quite at a Terabit scale. However SmartNICs support more general computations and can contain a few gigabytes of memory [3], making them beneficial to complement the coarse-grained query processing of programmable switches. Finally, end-hosts provide the largest memory capacity for storing longterm state, and the most flexible programming model, but can quickly become a bottleneck if performing expensive analysis on large numbers of packets. In this project we envision how to combine the scalability of programmable switches, the reasonably high-speed packet processing of SmartNICs, and the flexible processing capabilities and greatest storage capacity of host-based processing.

This combination of approaches is critical to support the large variety of attacks that target modern communication networks. There is a need to detect not only relatively crude volumetric attacks that overwhelm the network through sustained network activity (e.g., denial of service), but also more sophisticated attacks that probe for system weaknesses (*e.g.* SSH Brute forcing, port scan [4]), or attacks that exploit protocol dynamics (*e.g.* low-rate TCP attacks [5]).

Prior work has demonstrated how sketch data structures can be used to approximate flow-level statistics [6]. However, these approaches are only capable of monitoring the heaviest flows in the network. The data structures they use, such as NitroSketch [7], are focused on detecting heavy hitters, operating within the limits of the available amount of memory. However, their ability to detect low volume attacks that send only a few packets or slow attacks that send packets at low speed is more limited.

Our work seeks to harness heterogeneous data plane devices in order to build a network monitoring system that can efficiently and accurately observe the first set of packets for *all* flows in the network. Similar to previous work, we focus on out-of-band monitoring, i.e., monitoring that is performed on a mirrored stream of traffic which does not incur additional latency or impose bandwidth constraints on the real traffic. Nevertheless, our monitoring system must be carefully designed to ensure it can keep up with traffic rates of 100s of gigabits or Terabits per second.

This paper presents our vision for unifying programmable switches, network cards, and end hosts to acheive the goal of precise tracking of slow and low volume attacks. In Section II we demonstrate the need for fine-grained traffic monitoring through analysis of the MAWI traces [8], where we find that nearly 80% of incoming flows appear to be suspicious port scan attacks. While such a large number of flows sounds like it should be easy to detect, in practice they are often missed, as each flow is only 1-2 packets (representing less than 12% of the total packet volume). Next in Section III we discuss how to unify switches, SmartNICs, and hosts via a novel filter data structure on the switch, caching on the SmartNIC, and dataplane-based coordination protocols across all devices. This architecture reduces the load that must be sent to the host by up to 70X, while effectively detecting suspicious traffic at terrabit scale, as we show in Section IV.

II. Background and Motivation

A. Low and Slow Attack Types

While detecting (and preventing) crude volumetric attacks has traditionally been the focus of previous work, there are a number of low and slow attacks that involves sending a minimal, gradually paced stream of traffic aimed at depleting application or server resources. Different from large-scale DDoS attacks which can be detected quickly, low and slow attacks can hide themselves for long periods of time, all the while denying or slowing services to real users, making the detection and mitigation work more challenging.

Port Scan Attacks send a small number of SYN packets at low speed to explore targets for open ports that may indicate a weakness. Similarly, a ***SYN Flood*** tries to exhaust a target's kernel socket resources by exploiting the TCP 3-way handshake, while only requiring a small number of packets to be sent from the attacker.

HTTP POST/PUSH/PUT/GET Floods are layer 7 attacks that target URL endpoints and are known to consume significant resources, causing slowdowns or crashes in the victim server. These attacks all use standard URL requests, thus making it hard to differentiate them from valid traffic. Furthermore, their traffic volume is often under detection thresholds making detection even more challenging.

Slowloris and ***RUDY*** (also known as R U Dead Yet?) are attacks designed to hold open connections by continuously send partial HTTP GET or POST requests at a slow rate to keep sockets from closing. Compared to the HTTP attacks above, these attacks are purposefully *slow*, which both lowers the resources needed by the attacker and makes the attacks more difficult to distinguish due to the very small number of requests needed to have a large effect on the victim.

B. Slow Attacks in the Wild

To understand the current prevalence of these attacks on the Internet, we studied the flow characteristics of several publicly available traces. The first set we analyzed is the packet traces from the MAWI Working Group Traffic archive [8]. These traces are collected from the WIDE backbone in Japan.

MAWI: We present the analysis of a 64-minute trace, extracted from a 48-hour trace captured on April 13, 2022, in Table I. Of the nearly 90 million TCP flows identified in the 64-minute trace, we observed over 96% of the flows were no more than 2 packets (short flows). In fact, over 95% of the TCP flows were a single packet (1-packet flows). We also looked at those flows that had exactly two packets sent in the two directions - approximately 465K flows. When these flows have a SYN packet in one direction and either a SYN/ACK or a RST/ACK packet (essentially flows that do not complete the 3-way handshake), we classify them as half-open scans, indicating possibly an attacker generating a port scan attack. Just over 458K flows were such half-open scans, initiated by about 5,900 source IPs, which we then classify as malicious sources. We then determine the number of malicious flows (no more than 2 packets) initiated by these malicious sources is over 71 million flows (or about 79.8% of all the TCP flows). This clearly indicates a very high (and persistent) prevalence of very short flows (less than 3 packets), indicative of malicious traffic. Furthermore, the number of flows per malicious source host is heavy tailed, whereas the number of flows per legitimate host is quite small. There are very few legitimate sources that generate more than a handful of flows. We find similar traffic indicative of port scans in all the MAWI trace files during the last few years, with a generally rising trend (up to 83% malicious flows in a recent 2024 trace).

TABLE I: 64-minute MAWI trace from April 13, 2022.

Flow type	Number	Percentage (%)
TCP flows	89,266,373	100
Short flows	85,997,003	96.3
1-packet flows	85,201,286	95.4
Half-open scans	458,124	0.513
Malicious flows	71,223,022	79.8

TABLE II: Performance and resources for Tofino v1 Switch, Netronome Agilio LX NIC, and Intel i9 3.7 GHz host.

	Throughput	Memory	Programmability
Switch (per pipe)	1.6 Tbps	9 MB	P4
NIC (per port)	40 Gbps	4 GB	P4/MicroC
Host (per core)	10-24 Gbps	25 GB	Full

C. Heterogeneous Data Plane Devices

Table II characterizes the resources and performance of the heterogeneous data plane devices available in our testbed.

Switches: The Tofino v1 switch typically has two or four pipes, producing a total throughput up to 6.5 Tbps, with guaranteed line rate performance due to the strict pipeline based programming model. However, even the four pipe switch contains only 22MB of memory for the register blocks needed in stateful processing, and not all of it can be used by dataplane programs.

Although programmable switches have been used for traffic monitoring [1], [9], their limited SRAM memory for the program makes it difficult to retain state across packets and to hold the exact-match tables [9]. In fact, previous work [10], have also shown that due to resource limitations, standalone programmable switch solutions will suffer under high traffic, and defensive mechanisms can still be easily exploited. It often requires the monitor to focus on subsets of the traffic and use dynamic iterative refinement of queries processed at the programmable switch [1] to hone-in on the correct subset of traffic that contains the attack traffic. Limited access to the registers, limits on the number of match-action pipeline stages,

and memory limits place constraints on the amount of computation performed in a typical programmable P4Switch [11].
NICs: The Netronome Agilio LX SmartNIC that we use has two 40 Gbps ports and a total of 8GB of DRAM. While this card provides greater programmability than the switch, its bandwidth capacity is much lower.

Programmable SmartNICs have also been used for packet traffic monitoring [2], but they have their own limitations. Although they may outperform host-based solutions [2], effective traffic monitoring at Terabit speeds will require a relatively large number of such SmartNICs. Despite these limitations, SmartNICs can still nicely complement programmable switches with their larger memory. For example, DeepMatch [12] enables architecture-aware style programming of SmartNICs to support large volumes of traffic, and iPipe [13] can help reduce CPU loads on host. These designs complement our system to ensure flows with small number of packets do not get drowned out by heavy hitter flows.
Hosts: Network Function Virtualization software has developed ways for hosts to very efficiently process packets at high rates and low latency, often by bypassing the kernel and using shared memory to eliminate copying [14]. However, each core on our host running high performance DPDK software still can only handle about 10 Gbps when processing small packets (about 24 Gbps with large packets). Thus, processing terabit scale traffic would require a large number of hosts and most of their cores.

III. High Speed Monitoring Design

The primary challenge in integrating heterogeneous data plane devices is to mitigate the difference in forwarding rates, processing capabilities, and memory capacity between devices. Our core approach to resolving the challenge is threefold: *Firstly*, we propose switch-based filtering to significantly decrease the total number of packets transferred to the SmartNICs and hosts by identifying the benign flows that represent the majority of traffic in terms of packets. Our efficient switch-based data structure design leverages the massive bandwidth capabilities of the switch without its limited memory becoming a bottleneck.

Secondly, we are exploring how stateful dataplane processing on SmartNICs can alleviate the packet processing burden on hosts. Since SmartNICs have significantly more memory than switches, they can perform stateful aggregation and lightweight analysis before sending smaller amounts of traffic to hosts for more detailed analysis or long term preservation.

Thirdly, we seek to efficiently unify communication between these devices via data plane based protocols that allow for cross-device table updates that avoid the slow control plane paths. Control plane operations on the switch in particular can be dramatically slower than the dataplane, preventing multiple devices from working together for the dynamic stateful security analysis needed to catch low and slow attacks.

Figure 1 depicts the packet processing paths in our design. Packets first go through a *flow filter* on the switch, which efficiently determines if they belong to a flow marked as benign. If so, then no further analysis is needed, and the packet can simply be sent out of the switch. If the packet belongs to a flow not stored in the filter, it must be analyzed by the host to determine if it is suspicious. Instead of immediately sending the packet to the SmartNIC/host, its metadata is stored in a *flow log* retained on the switch. The logs are periodically flushed out to the SmartNIC in a batch to reduce packet transmission rates with negligible added delay.

The SmartNIC maintains a cache of currently active flows, but it does not have sufficient memory to track all flows over a long period of time. When the SmartNIC faces memory pressure, flows are evicted to the host, which has significantly more memory available to store flow state. Once the SmartNIC or host have gathered sufficient information about a flow to determine that it is benign, it sends an update message to the switch to add a new entry to the flow filter. This architecture limits the volume of traffic that must be processed at each layer of the hierarchy.

A. Switch-Based Filtering

Rather than attempting to use the switch for stateful processing, we instead view its role as a filter and traffic aggregator that will help reduce the load on and optimize communication to the host. To achieve this, our design requires a way to white list flows which have been deemed benign by the host or SmartNIC monitors. However, there may be millions of such flows in a Terabit scale network, and they will change over time. The (relatively) tiny amount of memory available on a switch cannot effectively track such a large list. Sketch-based filters such as Bloom Filters [15] and Cuckoo Filters [16] offer a good trade-off by significantly reducing the memory usage for set membership tests, with only a small sacrifice in accuracy. However, there are several challenges with implementing these filters on a P4 programmable switch, such as avoiding recirculations and maintaining low error rates.

While a standard Bloom Filter or Cuckoo Filter cannot have false negatives, this is no longer true once necessary adaptations are made for them to work in our environment – we would need the Bloom Filter to periodically replace old flows, and the Cuckoo Filter only avoids false negatives if it can grow in size once it is full, which is not feasible in a space constrained switch. In our scenario, false negatives correspond to flows being identified as needing analysis on the host even though we've previously marked them as safe. While these filters are often used in contexts where false positives are more acceptable than false negatives, that is not true in our monitoring context: a false positive potentially allows an attacker to bypass our system, while a false negative only leads to additional processing. Thus we propose a design that appropriatelys balances the risk caused by false positives and the overhead caused by false negatives.

Switch/Host Coordination and Flow Log Batching: In our design, the switch must coordinate with the host for two reasons. First, the switch must push information about incoming packets towards the host for analysis. Then, the host must add new entries to the switch's flow filter when it

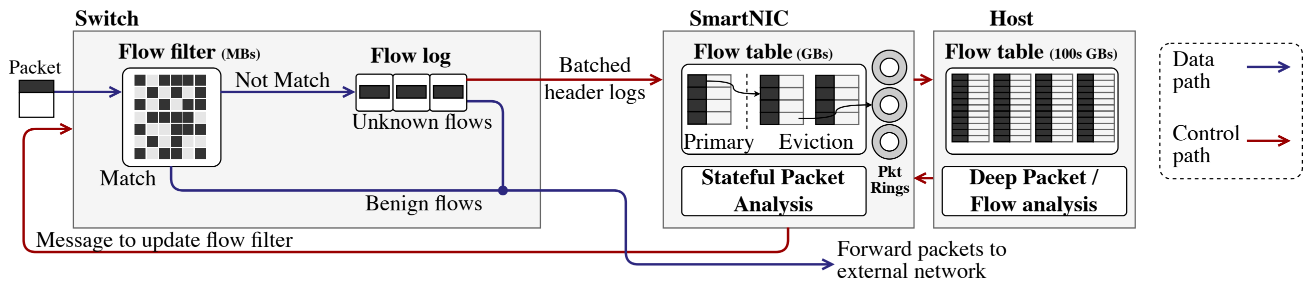


Fig. 1: Our switch-based Flow filter efficiently tracks the set of flows marked benign. Headers from unknown flows are batched by the switch before being sent to the SmartNIC for preliminary stateful analysis using flow state maintained in a multi-level Flow table. Finally, long term state is maintained on the host using its copious memory.

determines a flow is safe. Efficient real-time communication between the host and switch is crucial. The typical approach for reading or updating switch state is via its control plane interface; however, we find that its access rate is too low. To overcome this challenge, we propose a data plane based coordination scheme to increase this speed by several orders of magnitude. We optimize this process by using the host, as described in §III-B.

Switch $\rightarrow$ Host Packet Messages: We assume the host and SmartNIC run a security appliance to analyze packets at the start of all flows. With a straightforward design, every packet unknown to the switch must be sent to the SmartNIC or host. To mitigate the communication cost, we develop two mechanisms: *packet truncation* and *packet batching*. Firstly, rather than directly forward each packet in full, the switch truncates packets to just include key flow information from the packet header need for analysis. This can dramatically reduce the volume of data arriving at the host and SmartNIC. Secondly, to further reduce the number of packets that must be processed by the SmartNIC or host, we develop a packet batching mechanism, which buffers flow information from several packets in a *flow log*, and transmits it to the SmartNIC only when the flow log is full. Even a small buffer with space for a few packets can substantially reduce outgoing traffic by eliminating redundant protocol headers and unnecessary data due to frame padding. Also, because the buffer is extremely small, it incurs almost no latency to buffer the packets and send to the SmartNIC/host.

B. SmartNIC and Host-Based State Tracking

The host and SmartNIC in our unified dataplane typically have much more memory than the switch, thus allowing them to maintain state about a larger number of flows. The host in our design maintains a longer-term primary flow table, with information about all the flows that have been detected so far. We leverage the design in [17] for tracking flow state and logging flows using a simple 'flowcache' data structure on the SmartNIC, alongside the flow table on the host. Thus, a subset of these entries are cached on the SmartNIC's flowcache.

The SmartNIC packet processing engine updates the flow-cache by computing a hash of the 5-tuple in hardware. The SmartNIC memory (DRAM) can be multiple GBytes, potentially supporting from 25M to 100M flow entries. We use a two-level cache on the SmartNIC, with the first level (called the 'primary buffer') using a least-recently used (LRU) policy for evictions. The second level (called an eviction buffer) uses a least frequently used (LFU, packet count) based eviction policy, which favors large (elephant) flows to reside longer on the SmartNIC flowcache. This hybrid approach is effective at handling both a large number of recent flows (LRU) while also maintaining state related to elephant flows that might be bursty and are replaced by a large number of small flows in the LRU-based first-level cache.

Flows evicted from the eviction buffer of the flowcache are moved to a ring buffer from which we periodically flush snapshots of the flowcache to the host. Thus the host eventually maintains the full view of all of the flows being monitored. The cooperative monitoring by the switch, SmartNIC and the host enables our system to ensure minimal loss of flow information [17].

Because of the larger amount of memory on the SmartNIC, the flowcache on it can be much larger than what can be accommodated on the switch. Thus, by fully utilizing the SmartNIC's memory efficiently, we seek to collect additional flow level features. Beyond having access to some of the stateless packet level features (e.g. source port, destination port, protocol), which are contained in packet headers, the additional flow level features (e.g. flow size, inter-arrival time) monitored on the SmartNIC can provide higher accuracy when detecting attacks on the host. More specifically, when detecting for slow attacks (e.g. Slowloris), having the inter-arrival time is crucial for keeping track of the slowness between packets from a flow. In our design, the inter-arrival time feature can be obtained on the SmartNIC once the P4 Switch flushes each packet batch. The batching delay from the switch is negligible due to both the small batch size (< 5 packets) and the fact that inter-arrival time here can be relative time. As long as the batching delay stays consistent between each flush from the P4 Switch, then the change in time between each packet's arrival collected on the SmartNIC would also stay consistent. Thus, the use of this flow feature by the host would be accurate enough during the classification analysis.

As discussed previously, we have planned to introduce a monitor module on the SmartNIC and host that can run real-

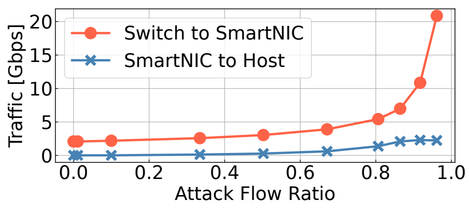


Fig. 2: An incoming traffic rate of 1.4 Tbps at the switch is reduced to 20Gbps on the SmartNIC and 4Gbps on the host.

time analysis on the packets received from the P4 switch. The monitor module will typically only need to look at the first few packets before making a decision about the flow (benign or suspicious). Once the SmartNIC or host has successfully analyzed the packets for the flow, then the information will be communicated to the P4 switch to update the flow filter with the packet's flow being marked as 'safe'. This communication uses the same batching mechanism to return a small set of updates in a single packet via the switch's dataplane, allowing it to update its Flow Filter rules without incurring control plane costs.

IV. Evaluation

In this section, we evaluate the effectiveness of our preliminary design through simulation and a prototype P4 switch and host combination in our testbed. The testbed evaluation shows how our system responds to synthetic traffic rates up to 1.4 Tbps. Our simulator implements the switch and hosts components of our design, and uses the PCAP traces from the MAWI project [8] to explore the characteristics of our design in more detail.

A. Terabit Scale Testbed Performance

We deploy our system on our testbed using a Tofino v1 switch, and a host with Netronome Agilio LX 40Gbps SmartNIC. We use a separate host and switch pair to generate packets, allowing us to create a configurable volume of traffic which is composed of a mix of benign and attack flows. This results in a maximum incoming load to the switch of 1.4 Tbps (170 Mpps). The traffic is then sent to the Tofino switch running our Filter code before reaching the SmartNIC and host. We configure the switch with a flow filter of size 4MB.

The traffic generator creates two types of flows: *benign flows* have 100 packets in total, each being 1024 bytes; *malicious flows* contain only a single SYN packet, such as from a Port Scan or SYN Flood attack. We adjust the traffic generator's parameter α from 0 (no attack flows) to 1 (all attack flows). Since we keep the total traffic volume constant at 170 MPPS, this means that the number of concurrent flows in the system will vary based on the attack rate. If there is no attack traffic, there will be an average of 1.7 M flows per second. It rises to 170 M flows per second if it is all attack traffic. Thus raising the attack rate also makes tracking flows within our system's limited switch memory more challenging.

Figure 2 shows the volume of traffic sent to the SmartNIC and thence to the host as we adjust the attack flow ratio, while maintaining a fixed 1.4 Tbps incoming rate to the switch. Even for large numbers of attack flows, the volume to the SmartNIC remains low, only passing 10 Gbps for $\alpha > 0.9$. The traffic volume to the host is even lower, showing the benefit of the SmartNIC cache. This traffic volume could be easily handled by a single core on a host (or even a fraction of a core). In total, the combination of switch and SmartNIC reduce the volume of traffic that needs to be analyzed on the host by more than 99.6%.

B. Trace-based Evaluation of Attack Monitoring

We next use our simulation of a Terabit/sec switch, focused on its monitoring data structures, to evaluate how our system can analyze real world traffic. We configure our system so that a flow is considered Safe once it has processed 7 packets, with smaller flows being treated as 'Suspicious'.

We analyze the ability of our system to track Port Scan attacks as an example of detecting low volume attacks, and compare it against approaches such as Count-Min Sketch and Sampling. We analyze an April 30, 2022 MAWI trace in which there are 9,319 concurrent flows per second or 34,063 packets per second, and consider how effective each system is at identifying the source IP addresses of port scanners. An attacker who sends only a small number of SYN probes will be harder to detect than one who sends many probes to different ports, so we evaluate different thresholds (10-400) corresponding to the minimum number of SYN probes a source IP must send to be considered as an attacker.

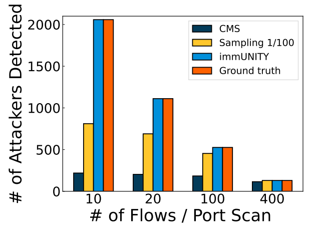
Here we compare against a Sampling approach, where only 1% of packets are analyzed by a flow table that has infinite capacity, and against Count-Min Sketch (CMS). We configure the sketch to keep counters based on hashing only the source-IP, and only give it SYN packets. Thus it attempts to count the number of SYN packets sent by each source. We then consider the top 50 such sources as potential attackers. Any subsequent flows that match a potential attacker are then stored in a flow table list for further analysis. This mimics the approach of other systems that use a sketch to detect heavy hitter flows and then store them in a flow table [7].

Figure 3a shows the number of attacks detected by each approach depending on the threshold of flows to be considered a Port Scan. Our design is able to perfectly match the ground truth data, correctly identifying all attacker source IPs. In contrast, the CMS-based approach does very poorly, missing between 10% to 90% of the attacks (Figure 3b). In this scenario we find that sampling performance still misses 60% of attackers when there is a low threshold.

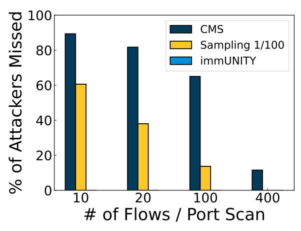
Overall, this illustrates the power of our system to precisely identify attack flows. While here we use it for Port Scans, since the attack analysis is performed on the host, a wide range of stealthy attacks can be handled without needing to revise the our system design. In contrast, a sketch based approach would need to be customized for each attack type.

C. Communication Costs & Switch Resource Consumption

Our design requires fast communication between the data plane devices in order to rapidly deliver packets for analysis



(a) Accuracy of detecting attackers



(b) Inefficiency in detecting attackers

Fig. 3: MAWI: port scan detection

and to update the flow filter. We measure the latency to send a 1KB packet from the switch to the host, parse the packet on the host, and then return a response (*e.g.* flow filter update) to the switch. We find that the median round trip latency ranges from 2.1 microseconds to 2.2 microseconds for rates between 10-39 Gbps (*i.e.* up to 4.5 million operations per second). This latency and update rate is dramatically faster than performing switch state updates via the control plane, which prior work has reported to take up to 1 millisecond, with a maximum throughput of around 100K updates per second [18].

V. Conclusions and Ongoing Work

Switches, SmartNICs, and hosts all have unique strengths and weaknesses that must be carefully combined to pinpoint unusual flows in the network. Switches excel at providing packet filtering, data extraction, and aggregation capabilities at terabit speeds. We propose to overcome their strict memory limitation by only maintaining state about benign flows, which our analysis of real traces shows make up a minority of flows, but the majority of transmitted packets. SmartNICs provide a first layer of stateful packet analysis, with sufficient memory capacity to track most active flows. We leverage a mix of LFU and LRU eviction policies to ensure their memory is used effectively, limiting the amount of data that must be propagated to the host. Finally, we need low-latency communication mechanisms between all of the devices; we leverage data-plane based communication that avoids the high overheads seen in control plane paths. Our design reduces the volume of traffic reaching a host by 99.6%, and more accurately detects slow attacks than sketch or sampling based approaches. We are continuing to optimize our system and expand its capabilities to intelligently classify multiple types of attacks.

Acknowledgements: This work was supported in part by NSF CNS-2210379, CNS-2210380, and NICT Contract No. 22401. Support for CAIDA's Internet Traces is provided by the National Science Foundation (CNS-2120399) and (OAC-2131987).

References

[1] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: query-driven streaming network telemetry," in *Proceedings of ACM SIGCOMM*, Aug. 2018, pp. 357–371. [Online]. Available: https://dl.acm.org/doi/10.1145/3230543.3230555

[2] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, "Turboflow: information rich flow record generation on commodity switches," in *Proceedings of ACM EuroSys*, Apr. 2018, pp. 1–16. [Online]. Available: https://doi.org/10.1145/3190508.3190558

[3] Netronome, "The joy of Micro-C," https://open-nfp.org/documents/48/the-joy-of-micro-c_fcjSfra.pdf, 2014.

[4] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE S&P*, May 2004, pp. 211–225. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1301325

[5] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proceedings of ACM SIGCOMM*, Aug. 2003, pp. 75–86. [Online]. Available: https://doi.org/10.1145/863955.863966

[6] H. Namkung, Z. Liu, D. Kim, V. Sekar, and P. Steenkiste, "SketchLib: Enabling efficient sketch-based monitoring on programmable switches," in *Proceedings of USENIX NSDI*, Apr. 2022, pp. 743–759. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/namkung

[7] Z. Liu, R. Ben-Basat, G. Einziger, Y. Kassner, V. Braverman, R. Friedman, and V. Sekar, "Nitrosketch: robust and general sketch-based monitoring in software switches," in *Proceedings of ACM SIGCOMM*, Aug. 2019, pp. 334–350. [Online]. Available: https://dl.acm.org/doi/10.1145/3341302.3342076

[8] WIDE project, "MAWI working group traffic archive," https://mawi.wide.ad.jp/mawi/.

[9] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proceedings of NDSS*, Feb. 2021. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_7C-2_24067_paper.pdf

[10] H. Zhou and G. Gu, "Cerberus: Enabling efficient and effective in-network monitoring on programmable switches," in *Proceedings of IEEE S&P*, May 2024, pp. 16–16. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00016

[11] K. Zhang, D. Zhuo, and A. Krishnamurthy, "Gallium: Automated software middlebox offloading to programmable switches," in *Proceedings ACM SIGCOMM*, Jul. 2020, pp. 283–295. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3387514.3405869

[12] J. Hypolite, J. Sonchack, S. Hershkop, N. Dautenhahn, A. DeHon, and J. M. Smith, "DeepMatch: practical deep packet inspection in the data plane using network processors," in *Proceedings of ACM CoNEXT*, Nov. 2020, pp. 336–350. [Online]. Available: https://dl.acm.org/doi/10.1145/3386367.3431290

[13] M. Liu, T. Cui, H. Schuh, A. Krishnamurthy, S. Peter, and K. Gupta, "Offloading distributed applications onto smartnics using ipipe," in *Proceedings of ACM SIGCOMM*, Aug. 2019, p. 318–333. [Online]. Available: https://doi.org/10.1145/3341302.3342079

[14] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. K. Ramakrishnan, and T. Wood, "OpenNetVM: A platform for high performance network service chains," in *Proceedings of ACM Workshop on HotMiddlebox*, Aug. 2016, pp. 26–31. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2940147.2940155

[15] M. Yoon, "Aging bloom filter with two active buffers for dynamic sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 134–138, 2010. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5066970

[16] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proceedings of ACM CoNEXT*, Dec. 2014, pp. 75–88. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2674005.2674994

[17] S. Panda, Y. Feng, S. G. Kulkarni, K. K. Ramakrishnan, N. Duffield, and L. N. Bhuyan, "SmartWatch: accurate traffic analysis and flow-state tracking for intrusion prevention using SmartNICs," in *Proceedings of ACM CoNEXT*, Dec. 2021, pp. 60–75. [Online]. Available: https://dl.acm.org/doi/10.1145/3485983.3494861

[18] T. Caiazzi, M. Scazzariello, and M. Chiesa, "Millions of low-latency state insertions on asic switches," *Proceedings of the ACM on Networking*, vol. 1, no. CoNEXT3, pp. 22:1–22:23, Nov. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3629144