Network Telemetry at the SmartNIC

Fall 2020 Independent Study Report

Hamza Motiwalla

December 17, 2020

Abstract

Large-scale data centers demand real-time network monitoring and management to ensure disruption-free service delivery. Monitoring and processing network packets in massive volumes requires minimal CPU or control plane intervention. In this report, we leverage SmartNICs to design and deploy line rate algorithms which allow network telemetry at the data plane without sacrificing bandwidth and ensuring low-latency packet transmissions. Employing linux kernel network capabilities such as eBPF and XDP, we design a robust network telemetry system capable of detecting network security threats at the SmartNIC.

1 Introduction

Network monitoring has traditionally relied on SNMP to gather packet flow statistics across static hardware configurations. With the advent of virtualization, network telemetry is no more as simple as setting up physical taps to monitor packet flow. Within data centers, packets travel from source to destination using underlay networks making it impossible to gather metadata for network analysis.

With the recent advances in network programmability, we can address the above issue by processing packets at the lowest level of the network stack. The extended Berkeley Packet Filter (eBPF) is an instruction set and execution environment inside the Linux kernel (Vieira et al. (2020)). eBPF provides us with a sand-boxed environment which enables modification, interaction and kernel programmability at runtime. eBPF coupled with XDP, the lowest layer of the Linux network stack, enables us to install eBPF programs into the Linux kernel to process packets. XDP allows us to hook/attach eBPF byte-code programs to specific network interfaces. Such an attached eBPF program is executed for every packet passing through the interface. Furthermore, we compile our program written in the restricted C language - to ensure safe kernel-level execution - into eBPF byte-code and XDP loads this code into the kernel or programmable SmartNIC at runtime (without recompiling the kernel).

To further expand on the Streaming Analytics Machine (Goodman and Grunwald (2019)), we implement the statistical primitives (such as TopK) as eBPF programs. After establishing a low-level eBPF foundation to collect packet data, we must make crucial design choices for our network telemetry system.

2 Related Works

Network monitoring is essential to not only debug performance impairments but also detect network vulnerabilities during attacks. To provide adequate support, network telemetry requires continuous real-time packet monitoring and analysis at the data plane. We propose using recent network programmability techniques such as eBPF and XDP to conduct real-time network telemetry. We also explore the recent advances in scalable network telemetry to gain insights into required primitives and potential design aspects.

Existing network telemetry solutions which exclusively utilize stream processors often suffer from low packet flow rates. To address this issue, Gupta et al. (2018) introduce Sonata which executes telemetry queries by partitioning each query across the stream processor and data plane. Sonata dynamically breaks down queries to execute majority of task on the network switch at line rate. While Sonata makes the best use of the limited data plane resources, it must incur performance overhead when executing rest of the query at the stream processor. HyperSight (Zhou et al. (2020)) proposes a declarative query language based on a stream processing model with a Bloom Filter Queue to provide in-network capability for packet behaviour monitoring. While HyperSight can monitor over 99% packet behavior changes, we aim to extract additional metadata from packets for advanced threat detection and analysis.

Another approach to real-time network analysis is in-band network telemetry where metadata is injected into packets to gather statistics. The P4 Language Consortium, defines in-band network telemetry as a framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane (Kim et al. (2016)). PINT (Ben Basat et al. (2020)) leverages P4 (Bosshart et al. (2014)) to encode the in-band network telemetry data across multiple packets. PINT is able to achieve impressive performance across applications such as congestion control and path tracing by injecting simply sixteen bits of metadata per packet. However, we take a different approach by refraining from altering the packets, since the injection and extraction of metadata bits hampers network performance as the volume of packets grows exponentially.

λ-NIC (Choi et al. (2019)) motivates offloading serverless computation to SmartNICs equipped with NPUs. In the same vein, Floem (Phothilimthana et al. (2018)) provides programming abstractions to offload computation, access packet fields, define queue mappings and even utilize the NIC as cache. Furthermore, Floem also provides a compiler to optimize execution between the NIC and CPU. Floem leverages NIC-offloading to improve the performance of a distributed real-time data analytics system by 75-96%. These research efforts motivate us to further offload network telemetry to SmartNICs in order to leverage their close proximity to the network to reduce overhead while computing network statistics.

3 Discussion

After implementing the streaming TopK algorithm (Golab et al. (2003)) as an eBPF program, we now discuss the next step towards developing a robust network telemetry system capable of flagging potential network security threats.

3.1 Netflows vs Raw packets

While SAL (Goodman and Grunwald (2019)) and FlowRadar (Li et al. (2016)) leverage netflows to compute telemetry statistics, we have the choice to use raw packets. Now that we have a foundation to collect packet information using eBPF code, we can either simply consume the packet by recording packet field values and passing them on to the control plane (ie host machine) via eBPF maps or we can create a new packet stream redirected to a specific host exclusively responsible for network telemetry.

3.2 Do we need stream processors?

After studying SAM (Goodman and Grunwald (2019)), Sonata and HyperSight which employ of stream processors such as Apache Flink (Foundation (a)) or Apache Spark (Foundation (b)), we debate the need for a stream processor for our application. Streaming network telemetry can provide better data point granularity and performance as compared to legacy methods which employ SNMP. The network size plays an important role in deciding the scale of our network telemetry application and whether the employing a stream processor justifies the overhead of query management and packet redirection.

3.3 Where do we compute the statistics?

After we have collected the required packet data using eBPF primitives, we must choose the ideal location to aggregate this data to compute the final results. One path is to write a powerful complex eBPF program to compute the results at line rate. However, this path comes with the limitation to restrict our eBPF byte-code size. The other path is to create a smart XDP loader capable of computing statistics and deciding the type and order of eBPF programs offloaded to the SmartNIC.

3.4 Available eBPF Primitives and more

While eBPF provides essential primitives required for efficient packet processing, we ponder upon the need for advanced primitives introduced as part of the eBPF kernel source code. Our TopK implementation ran into multiple obstacles due to the lack of basic primitives such as Most Frequently Used (MFU) hash map or a sort functionality for array maps. Therefore, we consider designing and implementing a strong suite of statistical eBPF primitives required for advance network telemetry. (Polylog streaming algorithms: Sum, Variance, TopK, K-medians, Quantiles, Rarity, Vector norms, Similarity, Count distinct items)

3.5 Employing Multicore SmartNICs

Multicore SmartNICs enabled with eBPF and XDP kernel features can be leveraged to implement some form of parallelism to boost performance. We could employ eBPF chaining where each packet must pass through multiple eBPF functions deployed on different cores (like a flow pipeline). The other way leverages the multicore functionality to dedicate one core for packet data collection while other cores share packet information using pinned eBPF maps and compute the required statistical results.

3.6 Current Industry Standards

Lastly, we must also look into the current eBPF program implementation deployed by companies such as Cloudflare, Cilium and Facebook to gather network telemetry results.

References

Ben Basat, R., Ramanathan, S., Li, Y., Antichi, G., Yu, M., and Mitzenmacher, M. (2020). Pint: Probabilistic in-band network telemetry. SIGCOMM '20, page 662–680, New York, NY, USA. Association for Computing Machinery.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95.

Choi, S., Shahbaz, M., Prabhakar, B., and Rosenblum, M. (2019). λ-nic: Interactive serverless compute on programmable smartnics. *CoRR*, abs/1909.11958.

Foundation, T. A. S. Apache flink — stateful computations over data streams. Blog.

Foundation, T. A. S. Apache spark - unified analytics engine for big data. Blog.

Golab, L., DeHaan, D., Demaine, E. D., Lopez-Ortiz, A., and Munro, J. I. (2003). Identifying frequent items in sliding windows over on-line packet streams. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, page 173–178, New York, NY, USA. Association for Computing Machinery.

Goodman, E. L. and Grunwald, D. (2019). A streaming analytics language for processing cyber data. *CoRR*, abs/1911.00815.

Gupta, A., Harrison, R., Canini, M., Feamster, N., Rexford, J., and Willinger, W. (2018). Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 357–371, New York, NY, USA. Association for Computing Machinery.

Kim, C., Bhide, P., Doe, E., Holbrook, H., Ghanwani, A., Daly, D., Hira, M., and Davie, B. (2016). In-band network telemetry. https://p4.org/assets/int-current-spec.pdf.

Li, Y., Miao, R., Kim, C., and Yu, M. (2016). Flowradar: A better netflow for data centers. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 311–324, Santa Clara, CA. USENIX Association.

Phothilimthana, P. M., Liu, M., Kaufmann, A., Peter, S., Bodik, R., and Anderson, T. (2018). Floem: A programming system for nic-accelerated network applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 663–679, USA. USENIX Association.

Vieira, M. A. M., Castanho, M. S., Pacífico, R. D. G., Santos, E. R. S., Júnior, E. P. M. C., and Vieira, L. F. M. (2020). Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. *ACM Comput. Surv.*, 53(1).

Zhou, Y., Bi, J., Yang, T., Gao, K., Cao, J., Zhang, D., Wang, Y., and Zhang, C. (2020). Hypersight: Towards scalable, high-coverage, and dynamic network monitoring queries. *IEEE Journal on Selected Areas in Communications*, 38(6):1147–1160.