Research Statement (2024)

Eric Keller

1 Introduction

We are seeing the scale of networked applications grow and the number of networked devices explode. This increased complexity stresses the underlying infrastructure, and those tasked with operating it, leading to performance, security, and cost challenges. It is my belief that the key to coping with this is a more programmable infrastructure. With more programmability, we can create abstractions and software systems which can capitalize on this increased flexibility with an ability to rapidly (and automatically) adapt the infrastructure to the changing conditions. My research has made (and will continue to make) significant contributions on both *enabling* and *capitalizing on* a more dynamic and programmable computing and network infrastructure, via such technologies as virtualization, software-defined networking, FPGAs, and the movement toward cloud based services. In this research statement, I highlight three thrusts I pursued during my Associate Professorship which exemplify my overall research[1]. In each, I discuss several published works as well as highlight some ongoing work.

2 Hardware/Software Co-Design of Network Processing

Central to the concept of increasing the programmability of the network itself, is introducing and enhancing programmability of the underlying platforms that perform the network processing. On the hardware side we have switches and network interface cards (NICs) that are now highly programmable, bringing flexibility to hardware's inherent high performance. In software, we have operating system and user level technology for faster packet processing, bringing performance to software's inherent flexibility. The key is leveraging each in the most effective way for the particular task. In this thrust, we focused on three key domains: (i) enhancing security through high-throughput telemetry with rich analysis, (ii) transparent acceleration of host-based network stacks, and (iii) enhancing resilience in host and rack-level network for large distributed deep learning applications.

2.1 High-throughput Telemetry with Rich Analytics for Active Security

As an example of leveraging progammability of the network to solve a critical problem, we introduced *active security*, a new methodology we pioneered [8] which introduces programmatic control within a novel feedback loop into the network defense infrastructure. The motivation behind active security is to streamline the security response and feedback mechanisms with minimal human interaction. A key challenge here is achieving rich processing capabilities across complex operations and complete telemetry information while doing so at high traffic rates (multi-terabit). In this work, we introduced a number of novel architectures exploring hardware software tradeoffs.

TurboFlow (Eurosys 2018) [22] (Best Student Paper) - Motivated by the desire for practical high coverage flow monitoring without sacrificing information richness, we introduced TurboFlow, a flow record (FR) generator optimized for programmable switches. Rather than trying to shoehorn flow record generation into either the programmable forwarding engine (PFE) or CPU, we decompose it into two complementary parts that are well suited for the individual processors. The PFE produces microflow records (mFRs) that summarize active flows over short timescales. Focusing on mFRs reduces the set of concurrently active flows to lower memory requirements and permits simpler data structures that map well to PFE hardware. A mFR aggregator, running on the switch CPU, stitches the mFRs together into complete flow records, using a key value data structure optimized to leverage

[1]Note: some works started at the the tail end of my Assistant Professorship and continued into the time period of my Associate Professorship.

performance features in modern CPUs. Together, the resulting design allows TurboFlow to support multi-terabit workloads on commodity programmable switches, enabling high coverage flow monitoring that is both information rich and cost effective.

***Flow (USENIX ATC 2018) [23] -** An important but unaddressed practical requirement in high speed networks is supporting concurrent network monitoring applications with diverse and potentially dynamic measurement objectives. We introduced *Flow, a switch accelerated telemetry system for efficient, concurrent, and dynamic measurement. *Flow places parts of the select and grouping logic that are common to all queries into a match+action pipeline in the PFE and exports a stream of records that software can compute a diverse range of custom streaming statistics from. There are two key innovations which we introduced in `*Flow` that makes it possible for this to work at high scales, without loss of information. First, we introduced a new record format for telemetry data, **Grouped Packet Vectors**, which compresses packet records without losing their information. It does this through de-duplication where a GPV contains a flow key, e.g., IP 5-tuple, and a variable-length list of packet feature tuples, e.g., timestamps and sizes, from a sequence of packets in that flow. From this, applications have complete information, and the GPVs reduce both the bandwidth and event rate, making it substantially more practical. Generating these GPVs in the data plane is challenging due to the variable length. The second innovation is a **Dynamic in-PFE Cache** that maps packets to GPVs at line rate in the PFE forwarding pipeline. The key here is that this introduced a line rate memory pool to support variable sized entries. Ultimately, dynamic memory allocation increases the average number of packet feature tuples that accumulate in a GPV before it needs to be evicted, and makes more efficient use of resources in the hardware.

JetStream (TNSM 2021) [20, 19] - To complement *Flow, we need to be able to process the flow records across different applications at high scale. For this, we introduced Jetstream, which uses a hardware-software co-design to efficiently analyze hundreds of millions of packets per second for multiple simultaneous applications allowing for network- wide, packet-level analytics without compromises. Our design is based on two key strategies. The first builds on the work in *Flow where PFEs export a stream of grouped packet vectors (GPVs) to software processors. We complement that with adding a load balancing component to the PFE to direct specific GPVs to specific NICs and programming the NICs to direct specific GPVs to specific cores. Our second strategy is to carefully optimize Jetstream's software component to exploit both the properties of network analytics workloads and our partitioning between hardware and software. Jetstream's analytics pipelines (which run application-specific logic) can be designed to operate independently of each other. This eliminates resource contention to improve both performance and scalability. Finally, guided by workload characteristics, we apply a series of domain-specific system optimizations. With this design, we showed for real applications that Jetstream is able to process between 5.4 and 15.9 million packets per core and scales linearly with increased cores (or between 86.4 and 254.4 million packets per second on a 16-core server). This represents a 184x and 24x throughput improvement over two state of the art systems.

eBPF and SmartNIC offload (IEEE NFV-SDN 2022) [4] (Best Paper) While some analytics tasks can be offloaded to programmable switches, ultimately, telemetry data needs to be processed by analytics applications in software. To reduce the resource footprint of software network analytics, we introduced a novel network monitoring primitive that consolidates logic which all monitoring applications require. The primitive can (partially) be offloaded to a SmartNIC and triggers applications only when required based on high-level traffic metrics, avoiding unnecessary and redundant computations. Our evaluation showed that the combination of conditional execution of analytics tasks and the use of modern packet I/O technologies (eBPF) not relying on expensive busy polling (as in DPDK) significantly reduces the resource footprint of performing continuous network analytics.

2.2 Transparent Network Acceleration

Software-based packet processing is widely adopted across a number of use cases, such as data-center load balancing, virtualized networking between containers, multi-cloud overlay networking, and 5G infrastructures. To overcome inefficiencies in the Linux networking stack, and enhance packet processing programmability, user space network processing toolkits, such as DPDK, have been introduced and are gaining in popularity. While we cannot question the high performance capabilities of the kernel bypass approach in the network functions world, we recognize that the Linux kernel provides a rich ecosystem with an efficient resource management and an effective resource sharing ability. In our work, we explored approaches to retain the rich capabilities of Linux, while accelerating the network stack.

Accelerating packet processing in Linux with LinuxFP (ICDCS 2024) [3, 1] - In this work, we introduced transparent acceleration into the Linux networking stack. To do so, we build on years of research in creating high-performance software-based packet processing systems. Rather than treating these technologies as alternative pipelines, we leverage the technology to create explicit fast paths in the Linux kernel. With this, Linux still serves as a complete implementation of all its supported protocols, but frequent operations on the critical path can be transparently handled by a fast path. We implement a controller that continuously introspects the Linux kernel to determine exactly what packet processing functionality is currently configured. The controller then synthesizes and deploys a minimal fast past into the packet processing pipeline that only implements functionality that is currently needed. In this way, common command line tools, such as brctl, control plane software, such as FRRouting (FRR), and higher-level management frameworks such as Kubernetes and Ansible, work without modification and transparently benefit from a faster network data plane. With this, we showed performance improvements over Linux for packet forwarding of 77%, without any modification to the management interface.

Accelerate the network TCP stack (IEEE NFV-SDN 2020) [2] - While LinuxFP accelerated the packet processing, we also looked to improve the performance of the TCP stack as well. We leveraged a high-performance user space TCP stack (mTCP) and recent (at the time) additions to the Linux kernel to propose a hybrid approach (kernel-user space) to accelerate SDN/NFV deployments leveraging services of the reliable transport layer (i.e., stateful middleboxes, Layer 7 network functions and applications). Our results show that this approach enables high- performance, high CPU efficiency, and enhanced integration with the kernel ecosystem. By having more efficient CPU usage, NFV applications can have more CPU cycles available to run the network functions and applications logic. We show that for a CPU intense application, using our system enabled it to have up to 64% more throughput than the previous implementation (with mTCP using DPDK).

2.3 Resilient Networks for Machine Learning

In ongoing work, we are continuing the theme of accelerating Linux networking and in this case capitalizing on an ability to transparently offload to a SmartNIC (that supports the Linux switchdev driver, which many do) to enhance the resilience of networks for distributed deep learning workloads. This is especially important due to two driving trends. The first is that there is a movement towards using Remote Direct Memory Access (RDMA) for lower latency and lower overhead communication. The second is that there is an ever increasing size of models and training datasets, thus requiring larger and larger networks (e.g., Facebook recently described their investment in an infrastructure with 24,000 GPUs). Together, these greatly increase the likelihood of failure while simultaneously increasing the cost of a failure. While there exists technology to handle failures of workers (at the application level) and in the datacenter network, the edge of the network (i.e., the host network port and NIC, and Top of Rack switches) does not have an adequate approach to dealing with failure.

Instead, failure at the network edge leads to communication errors and is assumed to be handled by application level fault handling. But, application level fault handling is not terribly graceful and leads to repeated work and extra training time. This is necessary for full host failures where the worker is actually unavailable, but when the host is working but unavailable due to network connectivity, this is wasteful and unnecessary. The problem is that leveraging redundancy at the network edge (host and rack level) with RDMA is a challenge. Since RDMA is inherently a hardware level protocol, it necessitates a programming model that is more closely aligned to the hardware. For applications to adequately handle failure, they would need to become more network aware to detect the failure and learn additional devices and paths, and be able to react to failure – all of which would be significant to support.

We are introducing a novel architecture for resilient host and rack networks that is completely transparent to applications and works with commodity hardware. This based on four core architectural principles: (i) Transparent, Hardware Offloaded Resilience, where we create port representors that applications bind to, and configure Linux networking with *tc* for directing from physical ports to this representor (and all of this gets offloaded to the NIC through the switchdev driver). (ii) Automated Host-level Path Selection, where we leverage standard data center protocols (BGP, EVPN, BFD) to learn and select network paths, and detect and react to failure. (iii) Avoidance of Soft Failures, where we assign each host with multiple IP addresses, detect path quality issues with explicit congestion notification (ECN) and RDMA counters, and use that to alter the packet to alter its ECMP path. (iv)

Scaling to large networks, where we introduce new algorithms to perform monitoring of the Linux kernel with provably correct convergence, and rule consolidation to accelerate time to configure. Preliminary results are showing that this can significantly improve the performance of distributed deep learning applications in the face of failure.

3 Elastic Resource Scaling in the Cloud

Cloud computing has rapidly expanded, both in the public cloud where cloud providers offer diverse services capable of supporting many types of applications, and in the private clouds where companies can manage their infrastructure more effectively. One of the key properties of cloud computing is scalability. In practice, providing fine-grained, dynamic, and elastic scaling in a cloud service is challenging due to the fact that limitations are inherited from lower levels: the OS is bounded by the machine, the container is bounded by the OS, and so on. In this line of research, we examine these boundaries and seek to decouple them towards more efficient scalability. In particular we look at these boundaries: (i) Network Function - (virtual) appliance, (ii) Container - OS, and (iii) OS - Hardware.

3.1 Stateless Network Functions

Network functions, such as firewalls, load balancers, and routers, are important components in every network by providing the ability to secure, monitor, and improve the efficiency of networks. While traditionally deployed as physical appliances, industry has recognized the need for more programmability and introduced the Network Functions Virtualization (NFV) movement (enabled and inspired by work from the academic community, including my own research during my PhD [17, 24, 15, 18, 16]). With NFV, network functions no longer have to run on proprietary hardware, but can run in software, on commodity servers, in a virtualized environment, with high throughput. Moving away from fixed physical appliances holds the promise that the network can achieve agility, but, without breaking the tight coupling of network functions to their underlying appliance form, this promise will go unmet. The central issue revolves around dealing with the state that is locked into the network functions. Network functions store state locally and use that as part of processing traffic. In tightly coupling the state and the processing, the elasticity, resilience, and ability to handle other challenges such as asymmetric / multipath routing and software updates becomes fundamentally limited.

In our research (NSDI 2017) [14, 13], we introduced a disaggregated architecture where, instead of maintaining state in the individual network functions themselves, the state is maintained separately and the network functions can access that state from anywhere and at any time through a well defined interface. The challenges include the fact that the processing of each packet/message would have an added latency, there can be a high rate of requests to the data store (per packet or message), and concurrent access to shared state across physical elements adds additional complexity. We overcame these challenges through a combination of leveraging modern technology from various domains (such as high performance computing) and domain specific optimizations (such as capitalizing on the nature of network traffic). We demonstrated that this proposed architecture is indeed possible, and we matched the throughput of other software-based solutions, while uniquely being able to seamlessly scale in and out (with zero packets dropped or connections broken), and instantaneously recover from failure (with no perceptible disruption) without relying on a complete duplication of every appliance on the network.

Stateless, Inc.: As a researcher, *impact* is the ultimate measure of success. Traditionally, this is measured in terms of number of publications (and citation counts), but I believe that impact is broader than that and can come in many forms. Along with the main Ph.D. student behind this research (Murad Kablan, who has since graduated), I have formed a company (Stateless) to commercialize this technology (licensed from the University of Colorado). Stateless is now 7 years old, received about \$1M in SBIR grants along with over \$20M in venture capital funding from premier VCs such as Foundry Group and Drive Capital. Our first product targeted data center operators and network service providers to equip them with a simple, scalable, and evolvable platform to offer new network services, such as connecting to and between public clouds. We have had numerous proof-of-concept and/or commercial deployments with some of the largest operators in the world. With the explosive growth of AI services, and the inherent need to move data around, we have recently launched a new product, AI Fabric, which leverages our technology to build a multi-cloud network fabric that is secure, cost effective, and performant. Companies can use AI Fabric simply and seamlessly to connect their data and applications to various AI services.

3.2 Event-driven, Sub-second Container Resource Allocation

Containerized infrastructure is quickly becoming a preferred method of deploying applications. In these deployments, per-container resources limits are used to prevent interference between containers and unchecked resource usage. Setting container resource limits is a trade-off between application performance and efficient use of underlying system resources – setting them too high leads to wasted / unused resources, setting them too low leads to performance issues (CPU throttling or Out of Memory (OOM) errors). Due to this trade-off, setting accurate limits is important and in practice, it is also difficult. Recent works set container CPU and memory limits by automatically scaling containers based on past resource usage. However, these systems are heavy weight and run on coarse-grained time scales, resulting in poor performance when predictions are incorrect.

We introduced Escra (ICDCS 2022 [7]), a container orchestrator that enables fine-grained, event-based resource allocation for a single container and distributed resource allocation to manage a collection of containers. Escra targets limitations in the API between the OS and container layers to provide fine-grained, near real-time adjustments to container resource limits for CPU and memory. Specifically, We expose fine-grained telemetry data from Linux's Completely Fair Scheduler (CFS). This allows Escra to quickly track and react to actual resource needs, resulting in both high performance (low latency and high throughput) *and* low cost (minimal slack). Further, we implement event-based memory scaling, which allows Escra to increase a container's memory upon an OOM event rather than allow the container to be killed. Using both microservice and serverless applications, we reduce application latency by up to 96% while increasing throughput up to 3.2x over a state of the art container orchestrator. These low latency and high throughput rates are achieved while simultaneously reducing the median CPU and memory slack by over 10x and 2.5x, respectively.

3.3 Distributed Node replicated Operating System

Even with Escra, we are still bound by the limitations imposed by the OS and machine – e.g., you can't allocate 8 CPUs to a container/process if the machine only has 2 CPUs. When developers need more resources than available on one server, they must design bespoke distributed systems or use distributed frameworks targeted at specific use cases. These approaches are specialized and hard to use, compared to programming applications for one host.

In ongoing work, we are introducing DiNOS (**Di**stributed **N**ode replicated **OS**), an operating system (OS) that spans multiple hosts and provides applications and processes with the abstraction of a single system image. With this approach of breaking down the barriers between the OS and the underlying hardware, the developer can access the aggregate resources of many hosts with the convenience and generality of the one-host programming model. In doing so, a POSIX-style process can spawn threads that run on different hosts, and it can allocate memory that comes from different hosts.

There is much prior work on realizing such a system, including work on cluster systems, distributed operating systems, and distributed shared memory. What distinguishes our work is the use of new hardware that provides sharing of host memory within a rack: a processor on one host can issue loads and stores of memory on another host, and these accesses are cache-coherent. This can be made possible with recent technologies, such as CXL. With that, realizing DiNOS brings a number of challenges. First, DiNOS must minimize the overheads that the kernel imposes on processes due to the distributed nature of the OS (e.g., synchronization and locking overhead when accessing kernel data structures, accesses to page tables, etc.). Second, remote memory accesses are slower and affect processes differently; DiNOS must support a wide range of CPU and memory scheduling policies including colocation of compute and data for processes that care. Third, DiNOS must isolate processes and the kernel from failures: if a host crashes, it should stop only processes that use resources from that host, not every process, and not the OS. Fourth, on the networking side, this raises new challenges with the underlying abstractions. What does it mean to have a socket that spans multiple hosts?

We are addressing these challenges, and believe we have a unique and novel architecture. One key contribution is a means to efficiently replicate kernel state between hosts. Another is introducing new networking abstractions for applications.

4 Enhancing Internet Security around the Complexities of Network Service Providers

In this research direction, we explore the broad topic of Internet Security, with a specific focus on unique complexities of network service providers. We first discuss the role of incentives in adoption of technology to secure the Internet, then present a line of research on the use of neural networks in traffic analysis, and how to evade that analysis.

4.1 Flipping Internet Security with a Focus on Incentives

Despite many advances in network security over the past couple of decades, the Internet continues to be plagued by security challenges for both consumers and the Internet as a whole. In this research direction, we investigated the role of incentivization (or lack thereof) as a primary driving factor for the adoption of security solutions.

Home Gateway (ACM CSUR 2023 [21]): We first conducted a retrospective assessment of consumer gateway security surrounding the role of network address translation (NAT), which we use to identify overarching trends, pitfalls, and missed opportunities for stronger security outcomes. We note that while a perimeter based security model afforded by NAT has never been a strong security approach, the simplicity, default-deny baseline behavior, and uniformity of design necessitated by address scarcity all served as strong incentives for deployment and use.

With the broad availability of addresses under IPv6, manufacturers are no longer bound by the default-deny design that NAT necessitated. Whether or not manufacturers are incentivized to continue offering a comparable default security baseline, and do so effectively, is unclear. To answer this question, we performed an assessment of IPv6 implementation found in ten consumer gateways. What we found is that many of the same security pitfalls surrounding NAT are being repeated, demonstrating that the need for security is not a strong incentive for actual implementation.

Internet Routing (ongoing): We then are considering what an incentivized approach to encourage security development and adoption could look like. For this we shift focus from the home gateway environment to the Border Gateway Protocol (BGP). Designed over thirty years ago, the Border Gateway Protocol (BGP) remains a ubiquitous and necessary protocol to support routing across the Internet. To address well documented security shortcomings, a number of approaches to strengthen BGP security have been proposed. One thing that stands out about each proposal is that they largely ignore incentives for deployment. The end result, unsurprisingly, is stagnant adoption.

In ongoing work, we believe the network research community needs to flip this problem around—we need to understand that network providers are a business first and build security solutions around that fact. From a business perspective, solutions that ease management or troubleshooting, enhance business value, or enable efficiencies all serve as strong incentives for adoption. To demonstrate this approach, we are building a real-time Internet routing database, named DND-Db (a Democratized Network Data Database). DND-Db serves as a foundational building block, which offers increased visibility for network administrators to manage, troubleshoot, and leverage network insights for business actions – key incentives for a business. In our initial prototype, we have been able to demonstrate that in adopting DnD-Db, a network provider can greatly enhance a key business objective – meeting service level agreements (SLAs) in the most cost effective manner. We also show that equivalent functionality to RPKI (route origin validation) and BGPsec (path validation) can be achieved, and even new capabilities can be seamlessly added.

4.2 Strengthening (or Evading) Traffic Analysis

Traffic analysis is important for network operators to understand threats in their network. At the same time, network providers can be subject to government agencies to monitor users of the network. An emerging means to perform analysis is through neural networks. We explore the vulnerability of such work, and how to make it more robust and practical. We then explore how we could evade analysis of the meta-data that is available to 5G network operators, for users to more safely use 5G networks.

4.2.1 Detecting Anomalies in Network Systems by Leveraging Neural Networks

Recent trends have given rise to network intrusion and detection systems (NIDS) built on neural networks. These systems can analyze traffic using packet or flow-level features, and can outperform (especially with unseen attacks) signature based analysis. However, it has been shown that neural networks are vulnerable to adversarial example attacks in other domains. These are small perturbations of the input that can bypass or purposely alter the classification. In the case of images, this might be changing a few pixels (imperceptible to the human eye) such that the classifier misclassifies a specific person as a different person or hides that person all together. In our work, we introduced foundational techniques for creating adversarial examples in network traffic, and making the neural network based NIDS more practical and more robust in an adversarial setting. Interestingly, at the time, this line of work seemed like an academic study, as doing this type of inference at high rates was not practical. Recent work on AI inference on programmable switches is showing that these can actually be done at terabit rates, which makes the impact of this work much more important now.

Evading Neural Network based NIDS (BigDAMA 2019 [9]): Generating adversarial examples for network traffic analysis is uniquely constrained by two key factors: (i) we must retain the network protocol correctness, and (ii) we must retain the attack's semantics. In our work, we introduced techniques to craft adversarial examples for networks by identifying traffic manipulations that can change the network features but remain within the constraints above. We demonstrated on existing NIDS and large network traffic datasets that this technique is highly effective.

Making more robust to adversarial examples (IEEE NFV-SDN 2020 [10]) The deterministic behavior of the previously proposed anomaly-based NIDS makes it easy to craft adversarial examples against them. We introduced Reconstruction from Partial Observation (RePO) [10] as a new method to build a more accurate NIDS which is also more robust in the presence of adversarial examples by utilizing denoising autoencoders and combining the inputs with multiple random masks before feeding them into the model. Our evaluation showed a 45% improvement in detection accuracy in an adversarial setting compared to other recently proposed systems.

Overcoming the challenge of training (IEEE TNSM 2022 [12, 11]): Deep learning models have shown their full potential in other tasks such as image classification, sentiment analysis, etc., when they were trained on labeled datasets. Training NIDS in a supervised manner is not a straightforward task. The problem is that labeling a huge dataset consisting of billions of packets is expensive, time-consuming and needs a human in the loop. Ideally, we would like to label a small portion of network traffic that includes some network attacks and be able to detect most types of attacks, including unseen attacks in the future, without a need to label them directly. We introduced Proportional Progressive Pseudo Labeling (PPPL) [12, 11], a domain adaptation technique that works across different input types (including network traffic), with much greater generality than existing techniques (which don't apply to network traffic well, or at all). Key to PPPL is that it tries to minimize the number of target samples that will align with a wrong class by excluding uncertain samples from the training set at the beginning of the training procedure and progressively bring them back into the training loop with a weight proportional to their certainty. Our experiments show that while PPPL is capable of improving the accuracy of image classifiers on visual domain adaptation tasks as good as state-of-the-art methods, it significantly outperforms them on other tasks with up to 62% improvement for anomaly detection in network traffic based on the F1 score.

4.2.2 5G Networks

Following from this line of work, in ongoing work, we are exploring the case of malicious network providers performing sophisticated traffic analysis (namely, in 5G networks). This can be a safety issue for individuals or organizations traveling in a foreign country where, for example, government agencies compel network operators to monitor users on the network. Data traffic is commonly encrypted, but meta-data is not. With this, we are exploring a set of technologies specific to evading traffic analysis in 5G networks – ranging from being able to swap identities (e.g., hardware identifiers) automatically, introduce personas (to make one user look like another), and activity shaping (coordination between multiple users to hide or fake an event). Also, we are ensuring the security of this through trusted execution environments, building on our past work on flexible secure hardware [6, 5]. This is part of the NSF Convergence Accelerator program, where we are exploring both the technical viability and the commercial viability (early indications are that there is strong commercial potential).

References

[1] M. Abranches, E. Hunhoff, R. Eswara, O. Michel, and E. Keller. LinuxFP: Transparently Accelerating Linux Networking. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2024.

[2] M. Abranches and E. Keller. A Userspace Transport Stack Doesn't Have to Mean Losing Linux Processing. In *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2020.

[3] M. Abranches, O. Michel, and E. Keller. Getting back what was lost in the era of high-speed software packet processing. In *ACM Workshop on Hot Topics in Networks (HotNets)*, 2022.

[4] M. Abranches, O. Michel, E. Keller, and S. Schmid. Efficient Network Monitoring Applications in the Kernel with eBPF and XDP. In *IEEE Conference on Network Functions Virtualization and Software-Defined Networking (IEEE NFV-SDN)*, Nov. 2021.

[5] A. Coughlin, G. Cusack, J. Wampler, E. Keller, and E. Wustrow. Breaking the Trust Dependence on Third Party Processes for Reconfigurable Secure Hardware. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, Feb. 2019.

[6] M. Coughlin, E. Keller, and E. Wustrow. Trusted Click: Overcoming Security Issues of NFV in the Cloud. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks &#38; Network Function Virtualization (SDN-NFV Sec)*, SDN-NFVSec '17, 2017.

[7] G. Cusack, M. Nazari, S. Goodarzy, E. Hunhoff, P. Oberai, E. Keller, E. Rozner, and R. Han. Escra: Event-driven, Sub-second Container Resource Allocation. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, July 2022.

[8] R. Hand, M. Ton, and E. Keller. Active security. In *Proc Workshop on Hot Topics in Networks (HotNets)*, 2013.

[9] M. Hashemi, G. Cusack, and E. Keller. Towards Evaluation of NIDSs in Adversarial Setting. In *ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks (Big-DAMA)*, Dec. 2019.

[10] M. J. Hashemi and E. Keller. Enhancing robustness against adversarial examples in network intrusion detection systems. In *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2020.

[11] M. J. Hashemi and E. Keller. General domain adaptation through proportional progressive pseudo labeling. In *IEEE International Conference on Big Data (BigData)*, Dec. 2020.

[12] M. J. Hashemi, E. Keller, and S. Tizpaz-Niari. Detecting unseen anomalies in network systems by leveraging neural networks. *IEEE Transactions on Network and Service Management (TNSM)*, 19(3), 2022.

[13] M. Kablan, A. Alsudais, E. Keller, and F. Le. Stateless Network Functions: Breaking the Tight Coupling of State and Processing. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, 2017.

[14] M. Kablan, B. Caldwell, R. Han, H. Jamjoom, and E. Keller. Stateless network functions. In *Proc. Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*, Aug. 2015.

[15] E. Keller and E. Green. Virtualizing the data plane through source code merging. In *Proc. Workshop on Programmable Routers for the Extensible Services of Tomorrow (PRESTO)*, Aug. 2008.

[16] E. Keller and J. Rexford. The 'Platform as a Service' model for networking. In *Proc. Internet Network Management Workshop and Workshop on Research in Enterprise Networking (INM/WREN)*, 2010.

[17] E. Keller, J. Rexford, and J. van der Merwe. Seamless BGP Migration with Router Grafting. In *Proc. Networked Systems Design and Implementation (NSDI)*, 2010.

[18] E. Keller, M. Yu, M. Caesar, and J. Rexford. Virtually Eliminating Router Bugs. In *Proc. International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2009.

[19] O. Michel, J. Sonchack, G. Cusack, M. Nazari, E. Keller, and J. M. Smith. Software packet-level network analytics at cloud scale. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, 18(1):597–610, 2021.

[20] O. Michel, J. Sonchack, E. Keller, and J. M. Smith. Packet-Level Analytics in Software without Compromises. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, Boston, MA, 2018.

[21] K. Olson, J. Wampler, and E. Keller. Doomed to repeat with ipv6? characterization of nat-centric security in soho routers. *ACM Computing Surveys*, 55(14s), July 2023.

[22] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith. Turboflow: Information Rich Flow Record Generation on Commodity Switches. In *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[23] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith. Scaling Hardware Accelerated Network Monitoring to Concurrent and Dynamic Queries With *Flow. In *2018 USENIX Annual Technical Conference (ATC)*, Boston, MA, 2018.

[24] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. In *Proc. ACM SIGCOMM*, 2008.