# Eric Keller's Research Statement (Aug. 2018)

## 1 Introduction

While society is enjoying the ever increasing ubiquity of networked applications, the challenges facing the underlying network infrastructure are likewise becoming ever more complex just trying to keep up. We are seeing the scale of applications grow and the number of devices explode, which is putting tremendous stress on those tasked with keeping the infrastructure up and running in an efficient and cost effective manner. At the same time, we are seeing the scale, diversity, and complexity of network attacks grow with no end in sight. This also is putting tremendous strain on the management of the underlying infrastructure.

My mission is to make even the most complex network infrastructures dead simple to manage. It is my belief that the key to 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. This directly addresses the challenges arising with scale and complexity of both applications and security threats. 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. Of course, my research has benefited tremendously from great collaborations and from inspiration from my fellow researchers in this field which have inspired and supported my work. In this research statement, I highlight three thrusts I pursued during my Assistant Professorship which exemplify my overall research.

## 2 Stateless Network Functions

A network function is a general term for any in-network processing, and has been known by other names historically such as a middlebox or a network appliance. These network functions are important components in every network infrastructure by providing the ability to secure, monitor, and improve the efficiency of networks. In fact, they are the main mechanism for network administrators to extend the functionality of their network. Some examples available today include firewalls, load balancers, and routers.

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 [8–11, 20]). 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 a foundational re-architecting of network functions, this promise will go unmet.

In this section, I describe the core challenge of making networks more agile (*i.e.*, why NFV and associated band-aids to work around its problems will fail) and how my research is solving this problem at the root. To emphasize the importance of this research, this solution is a fundamental and foundational contribution to this space, and I believe will transform how networks will be built. Toward this vision, I am commercializing the technology, which we also describe in this section.

## 2.1 Building a Better Network through Disaggregation

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.

Despite the positive advances in moving toward virtualization (decoupling hardware from software) and research efforts to work around the issues of state (*e.g.*, move around or checkpoint state), today's reality does not match a vision of a highly flexible, easy to manage network. The issue is that these approaches all operate under the single underlying assumption that state needs to be tightly coupled to a specific device, or said differently, that the architecture that worked for physical appliances should be retained for virtualized network functions

In our research [5–7], we stepped back and proposed that we can break the underlying assumptions. For this, 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. In doing so, we not only ease many of the issues that cause networks to be so brittle today, but also open new possibilities for transforming how network functions are architected and how networks are managed. The problem is that this seemed impossible (we were told this would never work by fellow researchers and people in industry when we first proposed it). 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.

## 2.2 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. In the case of this work, we believed the greatest path toward impact involves industry adoption, as production networks today are under tremendous pressure. 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). Beyond the research impact, I believe this is also important from the perspective of the University. Boulder has a strong entrepreneurial community, so having strong startups that come out of University research, and strong participation by faculty in the community, is important to ensure the University's role in this ecosystem.

**About Stateless:** Both my co-founder and I understood that it's not just "cool tech" that will make a successful company, but that the technology needs to solve a real problem companies are facing today (that is, there's a product market fit). We also understood that we have a great resource in the Boulder entrepreneurial community to help us out. We kicked off the process by participating in the CU New Venture Challenge where we won an award in the research category, and made our first public appearance as a company at the Boulder Beta event as part of Boulder Startup Week. From there, we participated in the NSF I-Corps program, where we conducted over 100 interviews of potential customers, and determined there is indeed a product market fit. We were then accepted to participate in Techstars, Boulder (Spring 2017). Techstars is an elite accelerator, which is run as a 3 month program where roughly 10 companies (out of thousands of applicants) all share an office, and receive daily education and mentoring on how to build a successful company. Coming out of that, we raised $1.5M from investors and received an NSF SBIR Phase I grant ($225k). We completed the Phase I project, and followed up by receiving a Phase II grant ($750k).

The current state of the company is that we have 13 employees, and a strong advisory board of 4 experienced professionals. Our product is in Beta (*i.e.,* pre-production quality, but usable to test the product), and being tested by multiple customers (including a Tier 1 provider)[1]. We have a full product release scheduled for March 2019, and we are currently starting the process to raise a Series A financing round, which will allow us to support these large deployments and expand to more companies.

**What this Enables:** What is particularly interesting is that what we have built is not only solving a technical problem companies currently have in updating their infrastructure, but also filling a substantial hole that has emerged in recent years due to a fundamental re-organization of enterprises' IT infrastructure. That is, we are now in the midst of a fundamental shift, from where infrastructure is built to meet an application's needs, to one where an underlying infrastructure is able to be shared and consumed on demand as a Service (*e.g.,* in a public cloud provider, such as Amazon AWS). This allows businesses to move at much greater speeds. As the movement to the cloud is maturing, we are seeing companies adopt a hybrid strategy where they maintain a footprint in multiple public cloud providers, retain some private (on premise) infrastructure, and host some infrastructure in a service provider. There is currently a large hole in trying to enable this highly dynamic interconnection fabric. This cannot be solved with legacy approaches, or with NFV solutions that are overly expensive and only minimally passable in meeting the needs of this use case, but can (and is) with the Stateless product.

# 3 Active Security

As an example of leveraging progammability of network to solve a critical problem, I have been researching *active security*, a new methodology we pioneered [4] 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; as a result, the cycle of monitoring via diverse sensors, hardening defense mechanisms, and responsive actions forms a high-rate OODA loop (**o**bserve, **o**rient, **d**ecide, and **a**ct). The OODA loop, described by John Boyd in 1976, presents a process for decision making in diverse environments where observations inform actions through a continuous feedback loop. Both defenders and attackers use OODA-like decision processes, the research challenge is ensuring that the defender's OODA loop cycles at a faster rate which provides a nimbleness that the attacker cannot match. Unfortunately, the OODA loops of today's defenders are currently operating at human-reaction timescales.

Here I describe my research which is leveraging programmable infrastructure for both increasing the fidelity and speed at which we're able to monitor and analyze network activity, as well as increasing this programmability of network devices to meet the needs of security applications. I then describe our vision of creating a software-defined security operating system, which abstracts the entire monitoring and control across networking, compute, and storage.

## 3.1 Network Analytics without Compromises

Reactive systems, as in Active Security, rely on both network monitoring and analytics for the reliable and secure operation of networking environments. A monitoring (or telemetry) system captures information about traffic, while an analytics system processes that information.

An ideal system needs full information about every packet in the network and full programmability of how that is processed, while doing so at the scale of modern data centers with terabits of traffic. Existing system compromise on this ideal, whether limiting the amount of information (*e.g.,* sampling packets), reducing the flexibility of how it is processed (*e.g.,* fixing some processing in hardware), or working only at small scales. In our research, we have been innovating solutions which allow us to not make compromises, consisting of both a lossless telemetry system telemetry system at high rates, and a high-performance and

[1] It should be noted that companies that provide infrastructure, such as ours, have a longer sales cycle than software services companies, so this is strong traction for this state of the company.

flexible stream processing system. In addition to academic venues, this work was recently presented at NANOG, the premier industry event for network operators, and was very well received.

#### 3.1.1 Lossless telemetry at high rates

We have been creating new approaches for telemetry which work at multi-terabit per second scales, without compromising on important practical requirements [15, 16, 18]. (This work was recognized with the best student paper award at Eurosys 2018 [16].) Most recently, we introduced `*Flow` [18], a practical programmable forwarding engine (PFE) hardware accelerated telemetry system that is not only flexible and efficient, but also supports concurrent measurement and dynamic queries. Our core insight is that concurrency and disruption challenges are caused by compiling *too much* of the measurement query to the PFE, and can be resolved without significant impact to performance by carefully lifting parts of it up to software.

`*Flow`'s design, plays to the strengths of both PFEs and servers. Instead of compiling entire queries to the PFE, `*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.

#### 3.1.2 High performance, flexible stream processing

Starting with the assumption that hardware can now send per-packet records (with *Flow) to the software-based analytics system, we then set out to address the limits of software analytics systems [12, 13].

The challenge in processing those records comes from the fact that: (i) modern cloud scale infrastructures run at traffic rates of several terabits of data per second, which can equate to hundreds of millions of packets per second, and (ii) modern analytics systems, specifically stream processing systems (which is an ideal programming model for network analytics), such as Apache Flink, simply are not capable of handling these rates.

Our insight is that stream processing for network analytics has fundamentally different characteristics than traditional uses of stream processing. In a paper discussing the preliminary results [13], we highlight some of these differences and show how they can impact the achievable processing rates. As a step towards validating our vision, we outlined an architecture and built a prototype with some preliminary optimizations. With this, our evaluations indicate more than two orders of magnitude speedup ($163\times$) over state-of-the-art solutions, with further optimizations to be introduced. Based on data from Facebook, we demonstrated that for an entire cluster in a Facebook data center, our prototype can analyze every packet leaving the cluster on a single commodity server (representing only 0.5% of the cluster's hardware). As a result, we believe that powerful packet-level analytics in software at cloud scale (without compromise) is indeed within reach. Ongoing work is going from prototype to full system, and evaluating on a wider collection of applications and traffic traces. We are also starting to engage with industry to get their input into new network analytic applications that they would like, which are enabled by this fundamental leap in both the information gathering with *Flow and ability to efficiently analyze with Jetstream.

### 3.2 Programmable Agents on Switches

The other side of reactive systems is re-programming the network to act on measured and analyzed information. Network security applications often require processing that is more advanced than software-defined network (SDN) data planes allow. Due to these limitations, SDN based security applications must implement much of their functionality in the *control plane* (*i.e.*, at the centralized network control server that manages the data plane switches). However, implementing functionality in the control plane hinders performance because the communication channel between the data plane and control plane is a bottleneck that adds latency and limits the amount of traffic that the security application can process. It also limits scalability because there are usually far fewer controllers than switches in a network. *For SDN to be practical for security applications, the data plane needs to support more advanced functionality.*

While some are taking a hardware approach (*i.e.,* add more functionality to the data plane ASICs), we introduced a new approach [14,17,19] that takes a middle ground between performance and deployability for SDN based security applications. Modern network switches have the general purpose hardware necessary to perform local packet processing in software, and even run operating systems with standard kernels, such as Open Network Linux. We introduced OFX (for OpenFlow eXtensions), a framework that allows an OpenFlow control application to easily load stack-independent data plane extension modules into OFX software agents running on dedicated OpenFlow switches that are already widely deployed. We showed how OFX extension modules can add security functionality to the network without requiring customized data plane hardware. And, based on our evaluation of real security applications in a testbed with a widely deployed model of a hardware OpenFlow switch (Pica 8 3290), we found that (i) overhead to process packets in an OFX module running on the switch was orders of magnitude lower than the overhead to process packets at the OpenFlow controller, and (ii) OFX provided similar benefits and functionality as previously proposed switch hardware extensions.

### 3.3 Software-defined Security Operating System

To complete the picture, we are currently building what we see as the "holy grail" for enterprise/cloud/data-center security management with software-defined infrastructure (SDI): a unified framework for security and management of disparate resources, ranging from processes to storage to networking. We have proposed S2OS [3] (software-defined security operating system) which offers an easy-to-use and programmable security model for monitoring and dynamically securing applications. S2OS is essentially an enterprise-level Security OS that abstracts security capabilities and primitives at both the host OS and network levels. S2OS facilitates infrastructure-wide security management with an easy programming interface for security administrators. S2OS will unlock a range of unprecedented security opportunities, including fine-grained, dynamic security programmability at entire infrastructure scale, information flow tracking across an entire data center, and easily translating simple, global security goals onto local OSes and network policy decisions.

## 4 Programmable Secure Hardware

Networked infrastructure not only needs more flexibility to handle increasing scale and complexity, but also to handle new threat models. For example, there is a trend towards outsourcing applications (including network functions) to a hosted environment (*i.e.*, 'the cloud'). This motivates the use of secure hardware to ensure the security and privacy of the application[2]. While our early work demonstrated the ability to secure NFV through Intel SGX [2], there is a fundamental problem with relying on fixed hardware solutions. Hardware implementations, like software, are also prone to implementation flaws, but due to the immutability of a hardware implementation, these flaws are much more difficult, if not impossible, to fix. Also problematic is that it is the hardware manufacturers which decide what features and capabilities the

[2]Alternatively, one could use fully homomorphic encryption, or some constrained variant, though these approaches are not practical.

hardware has, and this may not match the actual needs of applications, such as NFV (which are not fully known in advance).

With this, we have been innovating a new approach which enables *programmable* secure hardware. We leverage field-programmable gate arrays (FPGAs) to enable this, as there is an increasing ubiquity of FPGAs in infrastructures such as data centers (*e.g.,* both Amazon and Microsoft have publicly been integrating FPGAs into their data centers). Two initial key challenges which our research has been addressing include: how can we share the FPGAs for use in multi-application / multi-tenant environments, and how can we trust a reprogrammable system.

## 4.1 Sharing the FPGA between Dynamically Changing Applications

A central challenge in reaching our vision relates to how to share the FPGA between applications (which can be from different tenants) and the system. That is, we wish for multiple applications to be able to simultaneously use some of the FPGA's programmable fabric, while at the same time allowing the operating system to use some of the programmable fabric as well (*e.g.,* to connect to some I/O devices).

General run-time reconfiguration approaches have been proposed in the research community that would allow for sharing an FPGA, but, these approaches have been found to be impractical. We introduced a new approach [1], which overcomes these challenges. The key idea to enable this is to leverage a delivery model of a cloud deployment, in which we can effectively merge the modules into various static designs in the cloud, before delivery to the end user. We call this Cloud RTR (RTR for run-time reconfiguration). Specifically, our Cloud RTR system builds on the idea of "slots", or areas of the FPGA that can be reconfigured separately and shared between applications. To make this practical, where previous systems have failed, we provide a new approach to slot-based reconfiguration using a compilation system that abstracts away the underlying FPGA requirements. The resulting platform supports the use of slots at run-time, whereas previous systems only support slots at design time, and can share the FPGA between multiple parties.

## 4.2 Trusting Reconfigurable Secure Hardware

The second challenge that emerges in this is that the programmable nature of FPGAs raises a significant concern with regards to using them as a basis for realizing secure hardware – an attacker can read or modify the contents of the FPGA. This is in contrast to secure hardware systems built into silicon, which are "fixed", and cannot have their functionality changed after manufacture. We argue that the state-of-the-art approach, which involves some party encrypting and signing the configuration bitstream, doesn't solve the problem as it entirely relies on human / business processes to protect the keys. As history has shown with the frequent password and other data leaks (including secure boot keys), this cannot be counted on.

Instead, we are introducing a novel mechanism to break the trust dependence on third party processes for reconfigurable secure hardware [in submission]. We do this by putting the device itself in control over the programmability, thus removing the trust dependence on a third party's processes. This consists of two key aspects. The first is a self-provisioning mechanism where a device is initially brought up in a provisioning configuration, and then internally generates keys, and reprograms itself using these keys. In this way, the keys which control the configuration of the FPGA are only accessible internal to the device. The second is a policy driven update mechanism, where the hardware running in the FPGA is programmed with a policy which determines under what conditions to allow an update. In this way, we empower the secure hardware developer with the choice for how updates can occur (which could include a policy to block all updates). We demonstrated that this new mechanism is practical today with off-the-shelf FPGAs, and built an Intel SGX-like secure co-processor, that leverages both the updatable nature (*e.g.,* to overcome bugs), and the customizable nature (*e.g.,* a remote attestation capability that allows the device provisioner to choose who the root of trust is, rather than Intel's fixed root of trust being Intel).

## References


[1] M. Coughlin, A. Ismail, and E. Keller. Apps with Hardware: Enabling Run-time Architectural Customization in Smart Phones. In *USENIX Annual Technical Conference (ATC)*, Denver, CO, 2016.

[2] 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 & Network Function Virtualization (SDN-NFV Sec)*, SDN-NFVSec '17, 2017.

[3] G. Gu, H. Hu, E. Keller, Z. Lin, and D. E. Porter. Building a Security OS With Software Defined Infrastructure. In *Proceedings of the 8th Asia-Pacific Workshop on Systems (APSYS)*, 2017.

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

[5] 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.

[6] 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.

[7] M. Kablan, B. Caldwell, R. Han, H. Jamjoon, and E. Keller. Stateless Network Functions . In *Extended abstract and poster in the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2015.

[8] 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.

[9] 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.

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

[11] 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.

[12] O. Michel, J. Sonchack, A. J. Aviv, and E. Keller. Scalable Hardware-Accelerated Network Analytics. In *Extended abstract and poster in USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2018.

[13] 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.

[14] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith. (Poster) OFX: Enabling OpenFlow Extensions for Switch-Level Security Applications. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[15] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith. Accelerating Flow Collection on Commodity Switches. In *Extended abstract and poster in the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2017.

[16] 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.

[17] J. Sonchack, A. Dubey, A. J. Aviv, J. M. Smith, and E. Keller. Timing-based Reconnaissance and Defense in Software-defined Networks. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications (ACSAC)*, 2016.

[18] 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.

[19] J. Sonchek, A. J. Aviv, E. Keller, and J. M. Smith. Enabling Practical Software-defined Networking Security Applications with OFX. In *Proc. Network and Distributed System Security Symposium (NDSS)*, Feb. 2016.

[20] 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.