Eric Keller's Research Statement (Aug. 2015)

1 Introduction

I design and build secure and reliable networked systems using a cross-layer approach that draws from networking, operating systems, distributed systems, and computer architecture. My approach is to challenge existing assumptions – rather than solving a problem on top of the system, I look to change the system to make the problem go away fundamentally. With this, a cross-layer approach is central to my research as any given solution might straddle several of these areas.

My research introduces new systems, algorithms, and abstractions to enable a more manageable network and computing infrastructure. This is rooted in the fact that a significant portion of security and reliability issues are often a result of limitations in the management of networked systems. My research has been enabling and capitalizing on a more dynamic and programmable computing and network infrastructure, via such technologies as virtualization, software-defined networking, and the movement toward cloud based services. In the following sections, four main themes of my research are discussed with a sample of some of my work in each. These themes cover the entire spectrum – network device architecture (Section 2), network system management (Section 3), security management (Section 4), and network architecture (Section 5). Importantly, this work cannot be considered *my* research (alone), as I have collaborated with my advisors, peers, and students.

2 Overcoming the Rigidity Imposed by Network Device Architectures

Network functions such as firewalls, intrusion detection systems, and routers are central components of networks today. They are what provide the functionality that secures networks, monitors traffic, and ultimately enables the communication between end hosts to occur. Unfortunately, the design of these devices places undo impediments on network operators. As illustrated in Figure 1, we have been researching new device architectures and introducing new primitives to remove these impediments and help make networks more manageable, and in turn more reliable.

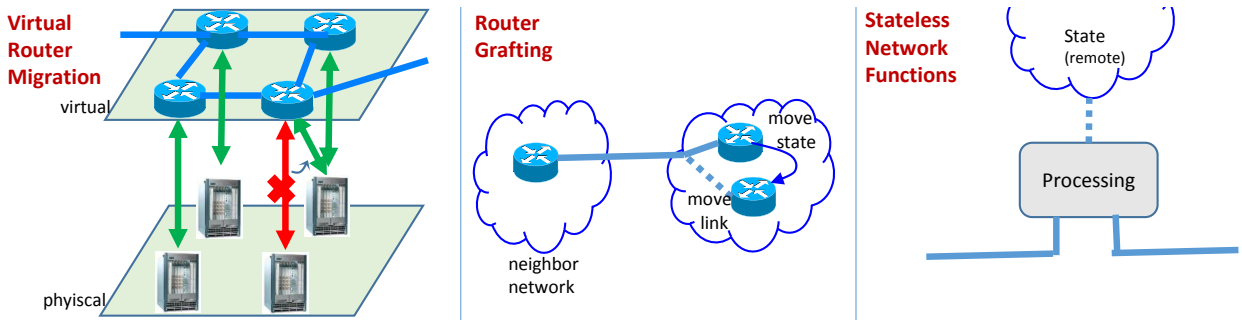

Figure 1: Illustration of virtual router migration, router grafting , and stateless network functions.

2.1 Router Migration and Router Grafting

Planned change is a fact of life for network operators who often need to repair faulty equipment, deploy new services, perform traffic engineering, and install new equipment to deal with the ever increasing traffic. Unfortunately, this change causes disruption, affecting the availability of cloud based services. Researchers have proposed extensions to the routing protocols to make change less disruptive (*e.g.,* by pre-calculating alternate paths). Of course, this suffers from deployment problems as it requires an Internet wide upgrade.

Instead, we note that many network-management problems stem from the same root cause – the monolithic view of a router where the hardware, software, and links are all considered one entity. Hence, we revisited the design of routers and proposed two new network-management primitives where (i) (virtual) routers are allowed to freely move from one physical router to another (VROOM) [23], and (ii) parts of a router can be seamlessly removed from one router and merged into another without any disruption (Router Grafting) [15]. As an example, today, the seemingly simple task of replacing a power supply, isn't simple at all – it requires great effort and coordination on the part of the network operators. Instead, with VROOM or Routing Grafting, we are able to keep the router operational during maintenance by (temporarily) migrating the entire virtual router to another physical router, or individual links and sessions to a collection of different physical routers.

We built and tested a router with these primitives through the use of virtualization to enable migration of the entire router, modifications to the hypervisor to enable our FPGA or software based data plane to remain active even during migration, and modifications to the guest operating system and a Quagga based BGP implementation to enable migrating a single link and associated BGP session of the router.

In addition to simplifying existing network-management tasks such as planned maintenance and service deployment, these primitives can also aid in handling emerging challenges such as reducing energy consumption. They can even be applied to traffic engineering, where we are now able to control where traffic enters the network rather than only being able to control the routes traffic takes within the network. The key question in applying these primitives is what to migrate and where to migrate it. For traffic engineering, the goal is to improve the utilization of the network (e.g., to minimize link congestion). We developed algorithms which we demonstrate through empirical evaluation and analytical formulation that dynamically re-homing customers can significantly improve the utilization of the available network bandwidth [16].

2.2 Stateless Network Functions

Going one step further, at the root of the challenge we were overcoming with link and router migration was dealing with the state within the router (and network functions in general). In our research [12], we are reexamining network function virtualization (NFV)[1] through a stateless design lens – inspired by the design of cloud-scale applications. Cloud applications are being decomposed into micro—dominantly *stateless*—services that rely on backend data stores (and middle-tier caching layers) to provide greater ability to easily scale up or down to match incoming demand, as well as better withstand failures. However, virtualized network functions remain stateful.

In our research, we are focusing on one primary question: *can the state of a network function be cleanly separated and persisted in a backend store or cache without loss of performance?* In decoupling the state from the processing of network functions, we can achieve more seamless elasticity and better tolerate failures for the individual devices.

Our initial system has demonstrated the ability to redesign network functions that decouple the state storage from the processing, and by leveraging modern, low-latency technology (namely the RAMCloud in-memory data store, and Infiniband low-latency transport medium), we were able to achieve performance within 10% of a native design (where state is coupled to the processing), but with much greater agility and ability to deal with failure. On going work is proving this out (i) at higher data rates, (ii) with a greater diversity of network functions, and (ii) at a larger operational scale.

[1] A new term introduced by industry to refer to the running of network devices (re-termed functions to be more general), such as firewalls or routers, in software on commodity servers rather than special purpose devices.

3 Introducing New Abstractions for Virtualized Infrastructures

Cloud computing is transforming the way people use computers and how networked services are run. The provider of a service (the cloud customer) is able to dynamically provision infrastructure to meet the current demand by leasing resources from a hosting company (the cloud infrastructure provider). The cloud infrastructure provider can leverage economies of scale to provide dynamic, on-demand infrastructure at a favorable cost. A key aspect of this is the virtualization abstraction provided. As illustrated in Figure 2, I have been researching new means to ensure security for tenants while maintaining the existing virtual computer abstraction, and introducing new abstractions for providing tenant control over the network while enabling providers to independently manage and optimize their physical infrastructure without disrupting the tenants' applications.

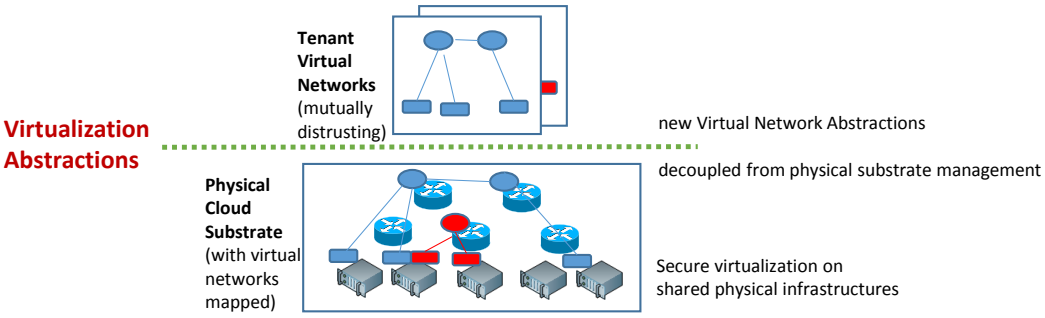

Figure 2: Illustration of the concepts related to virtualization abstractions. The cloud provider controls a physical substrate (switches and servers), tenants are specify an abstracted infrastructure, and the cloud provider maps and instantiates the tenant's virtual infrastructure.

3.1 Secure Cloud Computing without Changing the Virtual Machine abstraction

The multi-tenancy in cloud computing, where mutually distrusting customers lease resources from the same provider, underscores the need for a secure virtualization solution. The virtualization layer is quite complex and forms a very large trusted computing base that is not suitable for formal verification. A malicious VM can exploit these bugs to attack the virtualization software. Exploiting such an attack vector would give the attacker the ability to obstruct or access other virtual machines and therefore breach confidentiality, integrity, and availability of the other virtual machines' code or data. For cloud computing to gain widespread acceptance, this security concern needs to be addressed.

We did just that with our NoHype architecture where we proposed going to the extreme of removing the virtualization layer altogether, while retaining the abstraction and key features enabled by virtualization [17]. In contrast to previous work on hardening or minimizing the virtualization software, we eliminate the hypervisor attack surface by enabling the guest virtual machines to run natively on the underlying hardware while maintaining the ability to run multiple virtual machines concurrently. Our NoHype system that we designed, implemented, and evaluated on today's hardware [22], embodies four key ideas: (i) pre-allocation of CPU and memory resources, (ii) use of virtualized I/O devices, (iii) minor modifications to the guest OS to perform all system discovery during bootup, and (iv) avoiding indirection by bringing the guest virtual machine in more direct contact with the underlying hardware. Hence, no hypervisor is needed to allocate resources dynamically, emulate I/O devices, support system discovery after bootup, or map interrupts and other identifiers. NoHype capitalizes on the unique use model in cloud computing, where customers specify resource requirements ahead of time and providers offer a suite of guest OS kernels. At the same time, our system supports multiple tenants and capabilities commonly found in hosted cloud infrastructures.

3.2 New Virtual Network Abstractions and Management Technology

Initial cloud computing platforms have focused on virtualizing the server infrastructure, but largely ignored the network – they provide connectivity between virtual machines, but do not provide tenants with much control over the network. In our research, we have been researching various abstractions for providing control over a network, but that are decoupled from the physical infrastructure – making it so the customers only have to manage their services and the infrastructure provider has freedom to manage the physical infrastructure.

Single Big Router [14] : In many cases, what the cloud customer really cares about is being able to configure its specific policies and how packets are handled. As such, we proposed that they are provided with the view of a *single big router* that they can configure. Important to note is that we view a router in a broader sense than what they are traditionally thought of. They have many line cards to perform data plane functions, route processing blades which runs routing software to calculate paths through the network, general purpose processing blades allowing general software to be run (control or packet handling), and a switching fabric that connects all of the cards together. This platform will look very much like a router today, making it both familiar to network operators who are used to specifying routing policies in terms of router configuration, and generic enough to cover any type of in-network functionality. This single router abstraction inspired the research community to apply the concept to the emerging software-defined networking paradigm, and adopted the single big switch abstraction [6] – a concept that is having a lasting impact.

Virtual Software-defined Networks [8]: While the single big router or switch abstraction simplifies management, it also requires the cloud provider to make assumptions about the needs of the tenant (or completely over provision the physical infrastructure). As a second abstraction, we built off years of research on virtual network embedding (mostly theoretical), and created a virtual software-defined network abstraction for tenants. With this, tenants can fully specify an entire topology, which in turn serves as a means to communicate more fine grained requirements to the cloud provider. In contrast to an existing virtual software-defined network solution at the time [19], which provided the ability to divide physical resources, but did not virtualize the network, we provided full virtualization.

In either case, cloud customers manage virtual networks and the cloud provider needs to maintain that abstraction and still manage the physical substrate that the virtual networks run on. Virtual machine migration has proven to be an invaluable management tool that gives network administrators the flexibility to consolidate servers, balance load, perform maintenance, prepare for disasters, optimize user performance, provide high fault-tolerance, and reduce bandwidth usage without disrupting the applications.

We have been pioneering efforts to introduce migration capabilities for *entire* virtual (software-defined) networks [10, 13]. Our LIME architecture efficiently migrates an ensemble, a collection of virtual machines and virtual switches, for any arbitrary controller and end-host applications. To minimize performance disruptions, during the migration, LIME temporarily runs all or part of a virtual switch on multiple physical switches. Running a virtual switch on multiple physical switches must be done carefully to avoid compromising application correctness. To that end, LIME merges events, combines traffic statistics, and preserves consistency among multiple physical switches even across changes to the packet-handling rules. Using a formal model, we prove that migration under LIME is transparent to applications, *i.e.,* any execution of the controller and end-host applications during migration is a completely valid execution that could have taken place in a migration-free setting

We are continuing our research in creating new management primitives that can exploit this live migration capability. We previously demonstrated the ability to perform *inter-data center migration*. Currently, we are building a system for *datacenter defragmentation*. In cloud-based infrastructures, many tenants are creating, removing, expanding and contracting virtual networks. Over time, this leads to fragmentation of resources (*e.g.,* a tenant's virtual machines not being allocated on physical servers near one another), which in turn decreases the performance of the tenant's applications as well as placing extra stress on the cloud provider's infrastructure. Inspired by disk defragmentation (which moves around parts of files with the goal of making files occupy contiguous parts of disk for better performance when accessing them), we are performing data center defragmentation by periodically remapping virtual networks, using migration (as made possible with LIME) to transition to the new mapping.

4 Providing an Architectural (Speed) Advantage to Defenders

The task of securing an enterprise's, nation's, or military's cyber infrastructure is increasingly complex due to the diversity in attack vectors, sources, and targets. Today's tools to secure network and computing infrastructures are a mixed collection of individual components which can be configured, but have limited ability to learn from their environment and actively protect or seek out further information. In each case, most mechanisms available only have limited response measures (*e.g.*, updating the firewall to block traffic) and have static configurations that require human intervention to change any aspect of the configuration (such as altering what's being monitored or altering the IP address assignment to protect a current target).

To address these challenges, I have been researching *active security*, a new methodology we introduced [11] which introduces programmatic control within a novel feedback loop into the 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.

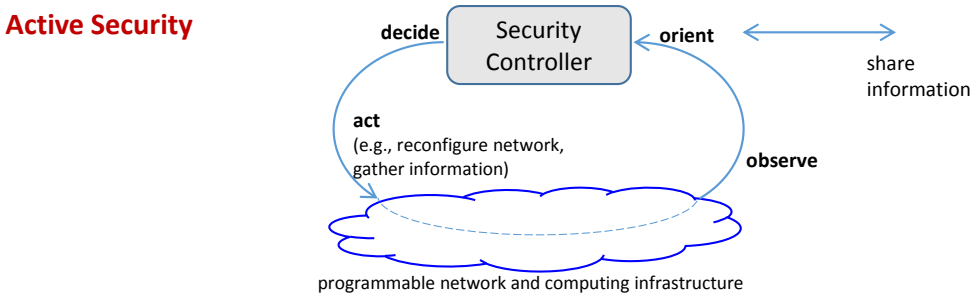

Figure 3: Illustration of the Active Security concept with an programmatic feedback loop which continuously collects information, dynamically re-programs the network and proactively seeks out new information, and shares information with others.

4.1 Enhanced Programmability and Automated Forensics

A programmable data network is integral in the active security approach as it give us the ability to dynamically modify the network and security infrastructure. But (i) we need to go beyond programmability, we need to better seek out and analyze data, and (ii) we need greater programmability than the defacto SDN standard, OpenFlow, is capable of providing.

Automated, In-Attack Forensics:

In the ideal case, defenses would prevent *ever* being compromised. Of course, that is a lofty and unrealistic goal and one where we believe security systems need to dynamically and automatically adapt to the changing context through our active security feedback loop. As such, an active area of research, and initial demonstration [11], is to perform in-attack forensics with automated analysis. Here we descrie a scenario that we created and utilized our active security in-attack forensics framework to perform analysis. Consider malware that has compromised a system and has begun to extract data. This may be caught through our intrusion detection system (IDS), and we may be able to limit the damage. Suppose, however, the malware also contained code to spread itself within the local area network, which is unmonitored by the IDS. In this case, through in-attack forensic evidence gathering, we are able to detect extra network sockets that

are open, and use this as input in the feedback loop, which will then trigger an action to reconfigure the network to block that traffic. The key here is again that we were able to perform information gathering actions on demand in order to get the information during an attack.

Software-defined Network Data plane extensions:

We are researching an approach 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 [4]. We are building 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 show how OFX extension modules can add security functionality to the network, inspired by AvantGuard [20], but without requiring customized data plane hardware. In preliminary experiments [21], our OFX extension module allowed networks using OpenFlow switches with standard hardware data planes to withstand over two orders of magnitude more attack traffic.

4.2 Crowd Sourced Sensing for Automated Classification

On the other side, we also need the ability to use the programmable infrastructure. The insufficient nature of static security configuration has received some attention from the research community in the form of highly programmable network infrastructures. To date, however, existing research has largely focused on the systems to enable a dynamic security infrastructure, but leave the automated use of these newly programmable infrastructures as an open topic. What is missing is how to define the classes of traffic dynamically, and how to handle the different classes of traffic as they are defined.

We have been researching this in the context of communication within a multi-tenant datacenter [5], and continue to purse automated classification of traffic based on observed behavior toward the goal of being able to actively secure networks. In our investigations, we argue that within a cloud environment, we can leverage the power of the crowd—the many interacting tenants—to achieve three primary goals: (1) focus isolation-related resources on tenants that more likely cause problems, (2) optimize communication between behaving parties, and (3) enable a security posture that automatically adapts to the dynamicity of tenants and services entering and leaving a cloud infrastructure.

We are building a general framework that defines simple, yet generic interfaces that can be easily integrated into different cloud management stacks and enable tenants to share information and use that information to automatically differentiate their various interactions—separating misbehaving tenants from well-behaved tenants.

5 Enabling Mainstream Adoption of Decentralized Services

Decentralized networked applications are known to provide great resilience and user control over data and privacy (ıthey are inherently more secure and reliable). Despite the benefits, the centralized (client-server) model has largely prevailed in the Internet today. Few decentralized applications have gained mainstream adoption (with the Internet itself being one of exceptions – imagine if the entire Internet was controlled and run by a single company). My research is looking to overcome the impediments that have prevented this mainstream adoption, which in turn will lead to networked systems that are more secure and reliable. This research is in the early stage.

5.1 Decentralized Ownership with Centralized Management

We argue that the proliferation of centralization is not because centralized applications are innately better, but that they are easier to use and manage. We have set out to combine the ease of use and management of centralized applications with the resilience and privacy benefits of decentralized systems. To do so, we

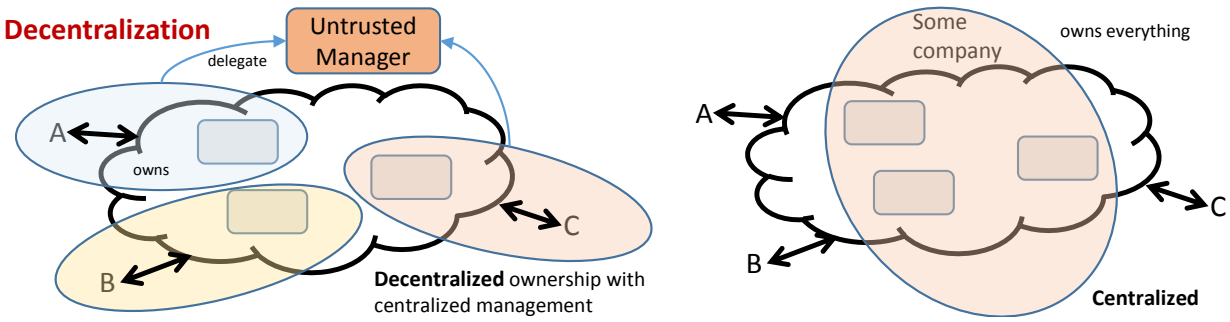

Figure 4: Illustration comparing decentralized networked systems where there is independent ownership, and centralized systems where there is a single entity which owns the entire infrastructure. Also illustrated is the delegation to an untrusted manager that we are introducing.

are introducing a new operational model, *untrusted delegation*, which enables decentralized ownership with centralized management. Users of decentralized applications following the untrusted delegation model may financially contribute by supplying some server hosting (cloud or virtual private servers), to the application while an untrusted manager takes care of administrative minutiae. We rely on three assumptions so that this will be possible: (i) the cloud provider enforces access control; (ii) in decentralized systems, an individual node is not trusted, but the community as a whole is trusted, which allows us to introduce a community verification approach that obviates any need to trust the central manager; and (iii) a central manager can facilitate the verification of user contributions, allowing for applications to define how contributions are rewarded and enforced. We have demonstrated how this model can be used to allow a user to deploy a Tor [7] relay node with no technical administration.

5.2 High-performance anonymity network with mutual Anonymity

In addition to researching the management of decentralized networked systems in general, we are also researching the architecture of the services themselves. In particular, as an initial example we are exploring a clean-slate anonymity network design. Anonymous communication is increasingly important as the Internet, one of the greatest inventions of our generation, was not designed with private communication as a first-order principle. We have seen tracking from agencies with far-reaching abilities such as the NSA, ISPs [1–3], and the websites we visit [18].

We are motivated by the limitations in the Tor anonymity network (the most widely used anonymity network). Namely, we have found there to be performance issues that impede the transfer of large sized content, and that it does not provide mutual anonymity (*i.e.*, anonymity for both sides), without extra hassle and extra performance penalties. At a high level, users must explicitly participate using a put/get style interface, where they simply put data into the network and the receiver gets the data from the network. Effectively, our architecture is a synthesis of three main existing concepts: onion routing for anonymous communication, the fully decentralized network authority model in Tahoe [24], and the use of access-controlled distributed information retrieval with required integrity checking in the named-data networking (NDN) model [9]. We are building and deploying this infrastructure and initial results are promising.

References

[1] `http://seekingalpha.com/article/29449-compete-ceo-isps-sell-clickstreams-for-5-a-month`.

[2] `http://www.dailytech.com/US+UK+ISPs+Track+Web+Habits+Sell+Data+to+Advertisers/article11391.htm`.

[3] http://www.cnet.com/news/verizon-draws-fire-for-monitoring-app-usage-browsing-habits/.

[4] Open network linux. http://opennetlinux.org, 2014.

[5] D. G. Andy Sayler, Eric Keller. Jobber: Automating inter-tenant trust in the cloud. In *Proc. Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2013.

[6] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker. Virtualizing the network forwarding plane. In *Proc. Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, 2010.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. USENIX Security Symposium*, 2004.

[8] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. *IEEE Internet Computing*, 17(2):20–27, Mar. 2013.

[9] C. Ghali, G. Tsudik, and E. Uzun. Network-layer trust in named-data networking. *SIGCOMM Comput. Commun. Rev.*, 44(5):12–19, Oct. 2014.

[10] S. Ghorbani, C. Schlesinger, M. Monaco, E. Keller, M. Caesar, J. Rexford, and D. Walker. Transparent, live migration of a software-defined network. In *Proc. ACM Symposium on Cloud Computing (SoCC)*, 2014.

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

[12] 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.

[13] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford. Live migration of an entire network (and its hosts). In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2012.

[14] 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.

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

[16] E. Keller, M. Schapira, and J. Rexford. Rehoming Edge Links for Better Traffic Engineering. *ACM SIGCOMM Computer Communication Review*, 42(2):65–71, 2012.

[17] E. Keller, J. Szefer, J. Rexford, and R. B. Lee. NoHype: Virtualized cloud infrastructure without the virtualization. In *Proc. International Symposium on Computer Architecture (ISCA)*, 2010.

[18] F. ROESNER, T. KOHNO, and D. WETHERALL. Detecting and defending against third-party tracking on the web. In *USENIX Conference on Network system design and implementation (NSDI)*, 2012.

[19] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, Oct 2010.

[20] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2013.

[21] 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.

[22] J. Szefer, E. Keller, R. B. Lee, and J. Rexford. Eliminating the Hypervisor Attack Surface for a More Secure Cloud. In *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2011.

[23] 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.

[24] Z. Wilcox-O'Hearn and B. Warner. Tahoe: The least-authority filesystem. In *Proceedings of the 4th ACM International Workshop on Storage Security and Survivability*, StorageSS '08, pages 21–26, New York, NY, USA, 2008. ACM.