Decoupling Policy from Configuration in Campus and Enterprise Networks

Nick Feamster, Ankur Nayak, Hyojoon Kim, Russell Clark
Yogesh Mundada, Anirudh Ramachandran, Mukarram bin Tariq
School of Computer Science, Georgia Tech

***Abstract*—This paper surveys our ongoing work on the use of software-defined networking to simplify two acute policy problems in campus and enterprise network operations: access control and information flow control. We describe how the current coupling of high-level policy with low-level configuration makes these problems challenging today. We describe the specific policy problems faced by campus and enterprise network operators; illustrate our approach, which leverages recent trends in separating the network's "control plane" from the data plane; and show how this approach can be applied to simplify these two enterprise network management tasks. We also describe our ongoing deployment efforts to build a campus network testbed where trial designs can be deployed and evaluated. We close with a summary of current and future research challenges for solving challenges within enterprise networks within the context of this new paradigm.**

I. Introduction

Network operators face many challenges in the process of maintaining high availability, good performance, and security. Many of the tasks that operators of campus and enterprise networks must perform involve complex debugging and policy specification tasks. For example, these operators may wish to allow only certain users access to various parts of the network; they may also aim to prevent certain sensitive data from "leaking" between different parts of the network, or from the internal network to the global Internet, or to rate-limit applications for certain users. Unfortunately, it is difficult for network operators to translate these types of high-level policy and design goals to implementation: until recently, network devices were largely "closed", only allowing for adjustment by means of a vendor's pre-defined configuration parameters. Worse yet, these configurations still typically occur at the level of individual devices, not based on a global perspective of the network.

This low-level configuration has made it increasingly difficult for network operators to configure networks correctly, as the size and complexity of these networks continue to increase. Although previous work has attempted to help network operators check the correctness of their configurations (*e.g.*, rcc [6]), configuring networks to achieve high-level tasks and policies remains a black art. Even when an operator finally manages to configure a collection of devices to achieve some high-level task or implement some policy, the configuration itself remains extremely brittle: because the configuration rests on the devices themselves and also depends on various low-level details (e.g., where the device is located in the network topology, software versions or vendor models of switches), a small change in configuration can result in an overall network configuration that fails to achieve the desired policy and is difficult to fix.

Two areas where this high-level problem is particularly acute are *access control* (defining who has access to what information and services on the network) and *information-flow control* (defining where on the network various information should be allowed to travel). Today, operators must achieve these tasks with a collection of firewall configurations, complicated VLAN configurations, and an agglomeration of low-level mechanisms on network devices, each of which are configured independently. The current state of affairs results in configurations that are difficult to modify, and nearly impossible to validate against some high-level policy.

Fortunately, recent developments in the design and capabilities of networked devices do offer some hope. In particular, many network devices have exposed programming interfaces to allow third-party software to directly control their behavior, effectively allowing networking control and logic to be defined in software ("software-defined networking"). A notable example of this paradigm is the OpenFlow switch specification [12], which allows network switches to be controlled from a remote, third-party device. This type of control-plane separation is a common theme across many designs and standards activities: the IETF FORCES working group and protocol standard [5] propose to separate the control plane from network devices; other similar paradigms exist (*e.g.*, Ethane [3], RCP [7], 4D [10]). Such a paradigm offers a wealth of opportunity for new, "clean" network designs for achieving critical tasks, because they move complex logic off of individual devices where control is proprietary, constrained, and federated into a logically centralized control platform.

In this paper, we describe how shifting control from individual network devices into a logically centralized control plane where policies can be expressed at a high level can result in much simpler configuration. For the two problems of information flow control and access control, we describe our ongoing efforts to explore how logically centralized network control and software-defined networking can simplify these exceedingly complex network operations tasks. We describe each problem at a high level, briefly discuss how they are solved in today's networks, and show how an approach based on a separate control plane can help network operators achieve these tasks in a simpler fashion. Effectively, this simpler operation results from the fact that network devices are no longer being configured one-by-one, but rather are being *directly controlled* in a coordinated fashion from a logically centralized point. We summarize two ongoing projects:

- The *Resonance* project, an ongoing effort to deploy a new network access control framework on the Georgia Tech campus network (Section II). A more detailed description of this work is also available in a workshop paper [11].
- The *Pedigree* project, an ongoing effort to develop an information-flow control system for enterprise networks (Section III). A demonstration of an early version of this system was recently presented [14].

These two examples illustrate the range of problems that a software-defined networking paradigm might be able to solve in a campus or enterprise network and point to a larger class of opportunities and challenges that software-defined networking presents for network configuration and management. This paper focuses on these two case studies, describes the current and planned deployment efforts, and discusses future opportunities both within the context of these problem areas and beyond.

The rest of the paper is organized as follows. Section II describes the campus network access control problem and how Resonance goes about solving this problem. We describe both the current design and open challenges. Section III describes the information flow control problem that many enterprise networks face, why this problem may be difficult to solve in practice, how software-defined networking approaches can simplify some information-flow control problems, and additional future challenges in this area. Section IV describes our current campus network deployment at Georgia Tech, where we are deploying and testing these new network designs in practice. We close in Section V with a discussion of general open research problems in this area.

II. Access Control

A common problem that enterprise network operators face is controlling who can access which portions of the network, and with which applications. For example, a network operator may wish to limit guests to only external, rate-limited HTTP Web access and grant employees more extensive access to the network. Ability to access the network may also depend on whether or not a host on the network is deemed to be infected. In today's networks, it is not possible to express these types of policies at a high level; rather, these high-level policies emerge from a collection of low-level devices, such as Web authentication portals, VLAN configurations, etc. Our ultimate goal is to make such high-level policy specification about access control possible. In this section, we elaborate on the high-level problem, our current approach, and ongoing research directions and open questions.

A. Problem Setup and Current Approach

Figure 1 shows the current START architecture [15], which is the authentication system deployed on the Georgia Tech campus. It is currently based on virtual LANs (VLANs) and VLAN Management Policy Server (VMPS) [17] and provides for dynamic network assignment that allows users to be placed on a separate network for authentication, scanning, and access to software update services to correct any problems discovered

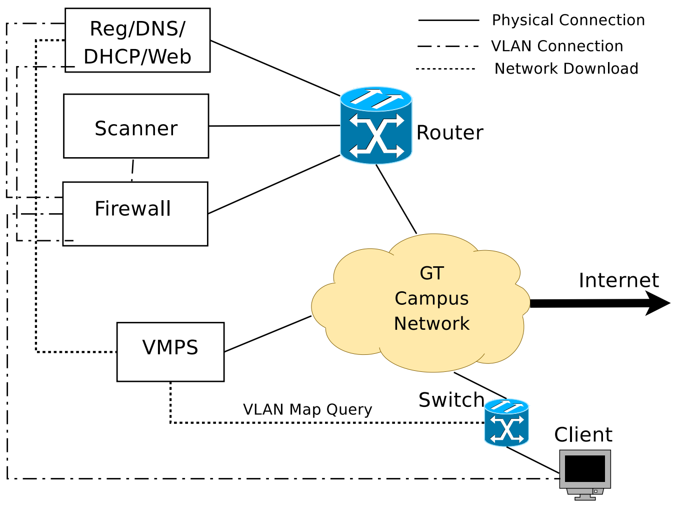

Fig. 1. Current architecture for START, the existing access control framework on the Georgia Tech campus network.

during the scan. After a client is authenticated and passes these tests, the system migrates a client to the regular VLAN with full network access and gives the client a public IP address. The START system supports the following functions:

The registration system provides the Web interface to the backend registration database, DHCP, DNS, authentication and updates for external systems. The Web interface guides users through the registration process. The DNS server for the network is a custom application written in Perl. It returns the IP address for the registration server for all DNS queries, except for a list of domains needed for updating workstations (*e.g.*, windowsupdate.com). Two instances of the DHCP server are run: One for the unregistered VLAN, and one for the registered VLAN. Each instance has its own configuration files that are created automatically from data in the registration system's database.

During registration, systems are scanned for known vulnerabilities. If the scan reveals vulnerabilities, the user is presented with these vulnerabilities and given an opportunity to update the system. The firewall for the network allows traffic to get to the appropriate update servers for Microsoft and Apple. The registration VLAN uses a firewall to block network traffic to unregistered desktops. However, the firewall allows Web and secure Web (*i.e.*, port 80 and 443) traffic to pass so that desktop machines can reach update sites. Various routers and switches are employed to facilitate creating the VLANs necessary for the needed networks. The local switches determine which VLAN for each machine that joins the network. The switch will download VLAN maps periodically from a VMPS. Unknown MAC addresses are assigned to the unregistered VLAN and known MAC addresses are placed onto the appropriate subnet. VMPS periodically downloads the VLAN maps from the registration server. Network security is enforced by creating ARP tables that map each MAC address to its registered IP and pushing that table to each router.

This approach has several shortcomings. First, access control is too coarse-grained. START deploys two different

VLANs to separate infected/compromised machines from healthy machines. This segregation results in all compromised hosts residing on a single VLAN; such a configuration does not provide proper isolation, since these infected hosts are not isolated from each other. Relying on VLANs makes the system inflexible and less configurable, because VLANs typically map hosts to network segments according to MAC address; the number of VLANs on a single network is also limited in practice, which constrains the number of unique policies any network might have. Second, moving individual hosts from one portion of the network to another is difficult. In the current configuration, when a machine is mapped to a different part of the network, it must be rebooted to ensure that it receives a public IP address, which is inconvenient because it relies on user intervention. Third, integrating access control with other systems that might help provide input for access control (*e.g.*, rate limiters, intrusion detection systems, etc.) is challenging, if not impossible—each new system introduces an additional "moving part" that increases complexity and makes the overall system less likely to function as intended.

B. *Resonance: Design and System Overview*

We are exploring how decoupling network control and configuration from the devices themselves can simplify network configuration and policy specification. Our first step has been to re-implement the START system in an OpenFlow [12]-based network design framework. The high-level idea is for a central *controller* to track the state of each individual device, associate each device with a user, and map each user-device combination to the appropriate corresponding security policy.

Each device's MAC address is associated with a user, and each MAC address is *also* associated with a particular state at any given time. Actions may arrive at the controller that might cause the controller to change the state of a particular device. For example, if a host or network intrusion detection system detects that a host may have been compromised, the controller may alter its view of the host's state and adjust networking forwarding behavior accordingly. It may even take different actions depending on the *type* of user is associated with the action. For example, if an intrusion detection system deems that a guest's machine is compromised, it may decide to completely quarantine that host from the network; if, on the other hand, the compromised host is deemed to belong to an employee or administrator, the controller may take less drastic actions (*e.g.*, filtering certain ports, redirecting the host to sites with software patches).

Currently, an operator specifies a policy in Resonance by defining a *state machine*, which reflects the complete set of possible states that a host might be in at any point in time, events that can result in state transitions, and the forwarding behavior that each switch should apply to hosts that are in each respective state. Figure 2 shows an example state machine that implements the basic policies for the Georgia Tech campus network. We have implemented and deployed this system on separate deployed network infrastructure running in our research lab; this network is currently supporting about ten concurrent users; Section IV describes this deployment in

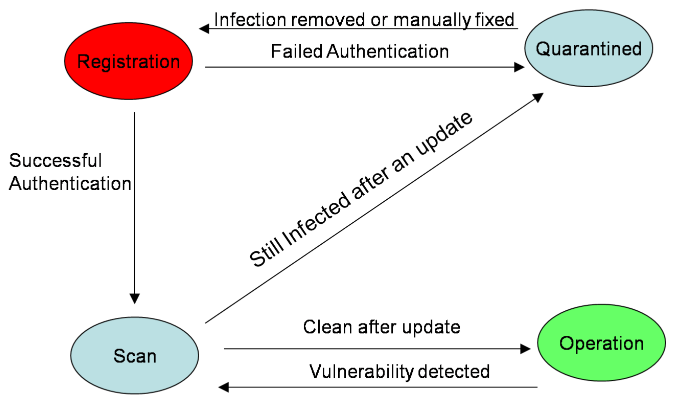

Fig. 2. State transitions for a host. The controller tracks the state of each host and updates the current state according to inputs from external sources (*e.g.*, network monitors).

further detail. Our future plans involve expanding the use of this network, as well as the range of policies that operators can express and implement on this network.

C. *Research Challenges*

Our work thus far on Resonance presents many avenues for future research directions. Perhaps one of the most interesting is the opportunity to define network behavior with a software controller, and the related opportunities and challenges that this presents. Resonance makes it possible to control network-wide behavior and define a wide range of policies in terms of a logically centralized software program, potentially making it possible for operators to define policies that are considerably more expressive and complex than with direct, low-level configuration of network devices. We are exploring the range of functions that the Resonance framework might enable, such as per-user rate limits and prioritization.

Another potential opportunity lies in the area of validation and testing: because Resonance allows network-wide policies to be expressed as programs at a controller, it may also be possible to apply conventional (and well-explored) software-engineering and testing techniques to network configuration. We are in the process of exploring how conventional approaches to software testing might improve network correctness and reliability, as well as how various programming languages could offer support to enable better support for checking and validating network operation.

III. Information Flow Control

We now describe our current efforts to decouple policy from network configuration for solving information flow control problems in the network. Information flow control problems in networking continue to be severe: In Deloitte's recent Global Security Survey, nearly half of the companies surveyed reported some internal security breach; of those, about a third of breaches resulted from viruses or malware, and another third resulted from insider fraud [4]. In this section, we describe a new method we are developing to control information flow in the network, rather than relying purely on host- or application-level control of information flow. We describe the problem setup, our initial design of an information-flow control system and how it might be implemented within a network based on OpenFlow [12], and describe various open research challenges in this area.

A. Problem Setup and Current Approach

Enterprise, military, and other networks may host sensitive information whose information flow must be controlled. Network operators must carefully control and monitor the flow of this information within and across the network. Today's mechanisms for controlling information flow have primarily been host-based: operating systems can "taint" portions of memory or applications based on the inputs to a particular process or resource. Unfortunately, if a host is compromised, or if an untrusted entity otherwise takes control of a host on the network, that information may propagate in unintended ways. Worse, once information has leaked, tracking the provenance of the leaked data to determine other hosts on the network that may have learned the information is particularly challenging.

Existing solutions to track and control information flow have taken two approaches: tracking taints on a single host (*i.e.*, tracking information flow across processes) or applying coarse features of network traffic that do not reflect any semantics about the traffic itself (*i.e.*, IP addresses and port numbers). These approaches result in complex (and typically imperfect) solutions because the mechanisms that are used to control information flow are added to the network on top of the existing infrastructure and at a limited number of monitoring or control points, it may be difficult to guarantee that information does not leak or propagate to unintended parts of the network.

Instead, we argue that the policy for controlling information flow should be integrated with the network layer itself. In particular, As systems become increasingly distributed, host and network-based mechanisms for controlling information flow should be synthesized to enable the flow of information to be tracked across the network. This mechanism would allow network operators to not only control how information propagates within and between networks, but also to devise more complex traffic control policies; for example, it might also be used to control which application traffic was allowed on which part of the network. With the appropriate mechanism, the information carried in the network traffic could also be attributed to a specific user in the enterprise network, as well as to a particular process. An operator could also use such a mechanism to allow certain privileged users to move information between different parts of the network (or between networks).

B. Pedigree: Design and System Overview

Pedigree is a mechanism for tracking information flow in networked systems by applying techniques from software taint analysis and machine learning to network traffic. Pedigree has two parts: A trusted *tagger* that resides on hosts and tags packets with information about their provenance (*i.e.*, identity and history of potential input from hosts and resources for the process that generated them); and an *arbiter*, which decides what to do with the traffic that carries certain tags. Pedigree allows operators to write information flow policies with expressive semantics that reflect properties of the actual process and user that generated the traffic.

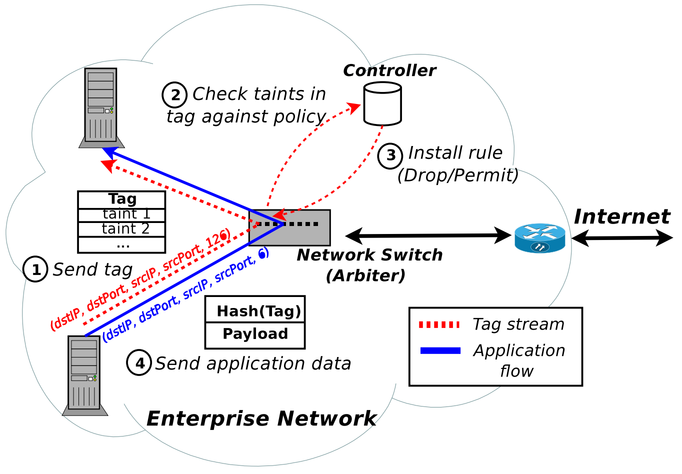

Fig. 3. Pedigree operation.

Figure 3 shows the basic Pedigree approach. Pedigree tracks interactions between resources (*i.e.*, files, processes, and sockets) in an operating system and attributes persistent tags to each resource. Pedigree annotates outgoing traffic with *tags*. When a process sends data on the network, Pedigree's tagger annotates outgoing packets with a tag that represent the provenance of a packet: essentially, the process that generated the traffic and where it has taken input from. When a process reads data from the network, the tagger updates the reading process's tags with tags on incoming packets. When a process wants to send data across the network, it communicates via an out-of-band control channel to the network controller, which decides whether to allow the requested network flows based on a network-wide policy. We are in the process of refining this design and deploying it in realistic enterprise-network settings.

C. Research Challenges

The design, implementation, and deployment of Pedigree entails many challenges. First, perhaps the most significant challenge is *how to specify and enforce policy*. Network operators need some intuitive language for expressing policies about how information should flow between various processes. Second, tainting must occur at the appropriate granularity without imposing unacceptable memory and performance overhead. The granularity must be fine enough so that individual resources can be tracked, and yet coarse enough to ensure that the system can track all of the resources on the host. Third, we must ensure that Pedigree imposes minimal overhead on application performance and is secure enough to withstand compromise and attack. Fourth, we must design the tags to fit in a packet and yet still contain enough information to carry meaningful semantics. We are grappling with many of these issues as part of our current deployment efforts (Section IV).

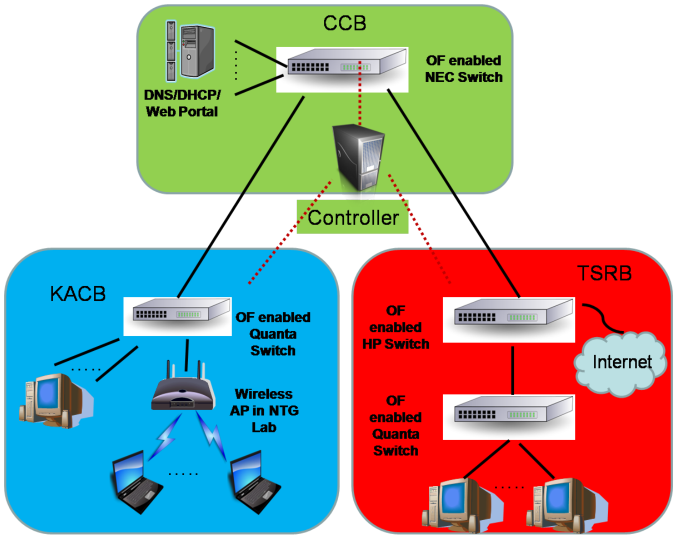

Fig. 4. Current testbed with connectivity between three buildings on the Georgia Tech campus network.

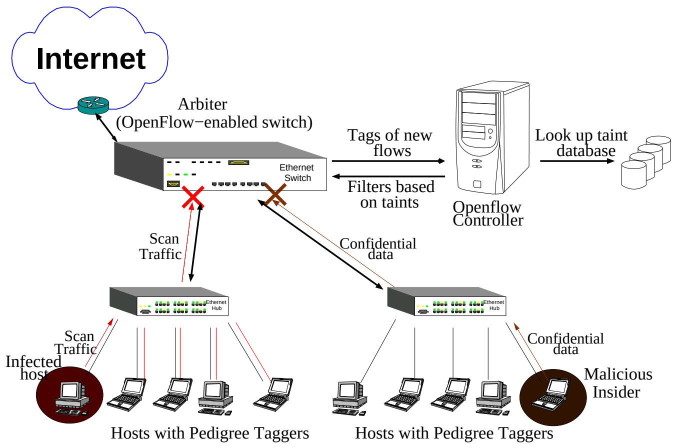

Fig. 5. Pedigree deployment in an enterprise network, where a malicious host is trying to scan its local network.

IV. Ongoing Deployment Efforts

We have deployed a testbed that supports our new network designs in three buildings at Georgia Tech. Figure 4 shows the current status of our deployment. The deployment is a dedicated network that is physically separate from the production network; it has its own IP prefix and upstream connectivity. This platform allows us to develop and test Resonance before deploying it on the production network. We have both wireless and wired connectivity on the testbed. Our current setup spans three buildings at Georgia Tech: Technology Square Research Building (TSRB), Klaus Advanced Computing Building (KACB) and College of Computing Building (CCB). We have OpenFlow-enabled 48$\times$1G switches from three different vendors: NEC, Toroki, and HP. Switches from Toroki are LB4G Quanta models with OpenFlow firmware version 0.8.9. NEC and HP switches both have 0.8.9 versions of the OpenFlow firmware. There is an access point in the networking lab to support wireless communication. As of now, the IP address space is currently taken from the GENI BGP-Mux deployment [8], [16].

All buildings are connected to each other using the NEC switch in CCB. Fiber paths connect both TSRB and KACB directly to CCB. Currently, all users connect to the network through the KACB building. In KACB, the OpenFlow-enabled Quanta switch is present in the data closet in the third floor. There are two connections coming out of the switch; one to the NEC switch and the other to the controller, both in CCB. Other ports on the Quanta switch are directly patched to ports in the networking lab. These ports in networking lab serve as entry points to the network. A wireless access point is connected to one of them and facilitates wireless connection in the lab.

Because the Quanta switches only support out-of-band configuration, we require a separate control path to the controller from the switch. For this purpose, a separate virtual LAN (VLAN) has been assigned from KACB to CCB, and from TSRB to CCB. Controller in CCB and Quanta switches in KACB and TSRB are part of this VLAN. All control traffic including initial OpenFlow handshakes between the switches and controller take place over this VLAN. Although out-of-band configuration allows better segregation of control and data traffic, yet for actual deployments such a configuration becomes infeasible since it necessitates the existence of separate VLAN paths from all switches in the entire network to the controller. On the CCB side, we have placed all management machines, including Web Portal machine, Controller machine and the DHCP/DNS server machine in the CCB machine room. All these management devices are connected directly to NEC switch in same rack. Connectivity with TSRB is still not fully functional. We have HP and Quanta switches in TSRB which are yet to be connected to controller in CCB. The RNOC switch, the gateway to the Internet for RNOC IP subnet, is also situated in TSRB, is waiting to be put online on the network.

A preliminary version of the Resonance system was demonstrated at the GENI Engineering Conference (GEC7) [9], and a preliminary version of Pedigree was demonstrated at *ACM SIGCOMM* [14]. Figure 5 shows the setup of our demo. We first set up many Pedigree-enabled end-hosts on a switched network. The switch supports the OpenFlow [12] standard and is capable of performing filtering decisions at high speed; this switch and its controller (on a different end-host) comprise Pedigree's arbiter. While being used for testing and developing these systems, the existing infrastructure can also be leveraged by other research projects for their own deployment using FlowVisor [1], which the ability to direct different control traffic to different controllers.

The campus network was recently upgraded to include approximately 275 switches that are capable of supporting the OpenFlow firmware. One of the more significant practical challenges in the campus deployment will be straining the scalability of the system on a production network without disrupting connectivity. For example, the proposed architecture may involve installing many flow table entries in the switches, which may either exhaust memory or slow lookup performance if entries are not stored efficiently, or if state is not offloaded to the controller. To address this concern, we will first stress-test the design on the research testbed and subsequently the architecture on a smaller number of production switches before completely rolling out the architecture.

V. Research Challenges and Opportunities

With proof-of-concept implementations and deployments of network systems that implement complex access control and information-flow control policies, we plan to turn our attention to how networks with a separate control framework can support a broader class of applications. The trends in our current research encourage us to reconsider the use of other various existing mechanisms and approaches to network design. In particular, the following general questions may pave the way for significant breakthroughs in the way that network operators configure their networks today:

Can network policy be decoupled from topology specification? In today's networks, operators use mechanisms such as virtual LANs (VLANs) to enforce logical separation between different parts of the network. Our work on designing access control mechanisms for the Georgia Tech campus network has shown that VLANs are typically too coarse of a mechanism for specifying policies, and that topology specification is an indirect way of implementing various policies.

How should complex network functions be factored between hardware and software, and between network devices and central control? As network policy begins to look less like low-level configuration and more like a software that can specify and implement complex logic, an important question arises regarding how this logic should be factored across the network infrastructure.

Conventionally, software has afforded operators and network designers considerable flexibility at the cost of data-plane forwarding performance. Recent advances, such as OpenFlow [12] and programmable network hardware [2] have begun to blur this distinction: OpenFlow permits software-defined control at a centralized controller, and programmable network hardware facilitates flexible processing of traffic in the data plane. This increased function creates the possibility of factoring complex logic between on-device network hardware, on-device "software exceptions", and centralized control. How various functions should be factored—and how this factoring should be specified—remains an open question.

Can software engineering and testing techniques play a role in improving the predictability, reliability, or security of networks? The software engineering and programming languages communities have spent considerable effort developing methods and techniques for testing the correctness of software programs. Software engineers face a problem of determining whether the a particular program will behave "correctly", often without a precise notion of what it means to be correct. Network operators face a very similar problem in configuring their networks.

Our research points to a trend where network policies look less like low-level configurations to instantiations and more like software programs. This trend may make various ideas from software testing (*e.g.*, differential testing [13]) more directly applicable. Similarly, programming languages that help programmers more directly reason about the correctness of a program's behavior may also help operators better reason about the complex behavior of networked systems.

Acknowledgments

This paper describes a broad research agenda encompassing several projects; further details are available in other early publications and technical reports publications [11], [14], [16]. This research is currently supported by an NSF CAREER Award 0643974, NSF awards CNS-0916732, CNS-0751134, and an contract from the GENI Project Office, "Campus Trials of Enterprise GENI". We acknowledge Ron Hutchins, Debbie Davis, Matt Sanders, and the Georgia Tech Office of Information Technology for their continued support of our deployment efforts on the Georgia Tech Campus Network.

References

[1] FlowVisor. http://www.openflowswitch.org/wk/index.php/FlowVisor.
[2] NetFPGA. http://www.netfpga.org.
[3] M. Casado, T. Koponen, D. Moon, and S. Shenker. Rethinking packet forwarding hardware. In *Proc. Seventh ACM SIGCOMM HotNets Workshop*, Nov. 2008.
[4] Deliotte. Global Security Survey, May 2009. http://www.deloitte.com/view/en_SK/sk/insights/deloitteresearch/article/8162586731101210VgnVCM100000ba42f00aRCRD.htm.
[5] A. Doria, R. Haas, J. Salim, H. Khosravi, and W. M. Wang. *ForCES Protocol Specification*. Internet Engineering Task Force, Mar. 2007. Internet Draft (http://www.ietf.org/internet-drafts/draft-ietf-corces-protocol-09.txt). Work in progress, expires August 2007.
[6] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, May 2005.
[7] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe. The case for separating routing from routers. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Portland, OR, Sept. 2004.
[8] N. Feamster, V. Valancius, and Y. Mundada. Bringing experimenters and external connectivity to GENI a.k.a. BGPMux, DTunnels. http://groups.geni.net/geni/wiki/BGPMux.
[9] 7th GENI Engineering Conference (GEC7). http://www.geni.net/?p=1626, Mar. 2010.
[10] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *ACM Computer Communications Review*, 35(5):41–54, 2005.
[11] A. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance: Dynamic access control in enterprise networks. In *Proc. Workshop: Research on Enterprise Networking*, Barcelona, Spain, Aug. 2009.
[12] OpenFlow Switch Consortium. http://www.openflowswitch.org/, 2008.
[13] A. Orso and T. Xie. BERT: Behavioral regression testing. In *Proceedings of the 2008 international workshop on dynamic analysis*, pages 36–42. ACM, 2008.
[14] A. Ramachandran, Y. Mundada, M. bin Tariq, and N. Feamster. Securing Enterprise Networks with Traffic Tainting, Aug. 2009. SIGCOMM 2009 Demo Session.
[15] Scanning Technology for Automated Registration, Repair and Response Tasks. https://start.gatech.edu/.
[16] V. Valancius, N. Feamster, J. Rexford, and A. Nakao. Wide-Area Route Control for Distributed Services. In *Proc. USENIX Annual Technical Conference*, Boston, MA, June 2010.
[17] Configuring Dynamic Port VLAN Membership with VMPS. http://www.cisco.com/univercd/cc/td/doc/product/lan/cat5000/rel_4_2/config/vmps.htm.