Improving Network Management with Software Defined Networking

Hyojoon Kim and Nick Feamster, Georgia Institute of Technology

ABSTRACT

Network management is challenging. To operate, maintain, and secure a communication network, network operators must grapple with low-level vendor-specific configuration to implement complex high-level network policies. Despite many previous proposals to make networks easier to manage, many solutions to network management problems amount to stop-gap solutions because of the difficulty of changing the underlying infrastructure. The rigidity of the underlying infrastructure presents few possibilities for innovation or improvement, since network devices have generally been closed, proprietary, and vertically integrated. A new paradigm in networking, software defined networking (SDN), advocates separating the data plane and the control plane, making network switches in the data plane simple packet forwarding devices and leaving a logically centralized software program to control the behavior of the entire network. SDN introduces new possibilities for network management and configuration methods. In this article, we identify problems with the current state-of-the-art network configuration and management mechanisms and introduce mechanisms to improve various aspects of network management. We focus on three problems in network management: enabling frequent changes to network conditions and state, providing support for network configuration in a high-level language, and providing better visibility and control over tasks for performing network diagnosis and troubleshooting. The technologies we describe enable network operators to implement a wide range of network policies in a high-level policy language and easily determine sources of performance problems. In addition to the systems themselves, we describe various prototype deployments in campus and home networks that demonstrate how SDN can improve common network management tasks.

INTRODUCTION

Computer networks are dynamic and complex; unsurprisingly, as a result, configuring and managing them continues to be challenging. These networks typically comprise a large number of switches, routers, firewalls, and numerous types of middleboxes with many types of events occurring simultaneously. Network operators are responsible for configuring the network to enforce various high-level policies, and to respond to the wide range of network events (e.g., traffic shifts, intrusions) that may occur. Network configuration remains incredibly difficult because implementing these high-level policies requires specifying them in terms of distributed low-level configuration. Today's networks provide little or no mechanism for automatically responding to the wide range of events that may occur.

Today, network operators must implement increasingly sophisticated policies and complex tasks with a limited and highly constrained set of low-level device configuration commands in a command line interface (CLI) environment. Not only are network policies low-level, they are also not well equipped to react to continually changing network conditions. State-of-the-art network configuration methods can implement a network policy that deals with a single snapshot of the network state. However, network state changes continually, and operators must manually adjust network configuration in response to changing network conditions. Due to this limitation, operators use external tools, or even build ad hoc scripts to dynamically reconfigure network devices when events occur. As a result, configuration changes are frequent and unwieldy, leading to frequent misconfigurations [8].

Network operators need better ways to configure and manage their networks. Unfortunately, today's networks typically involve integration and interconnection of many proprietary, vertically integrated devices. This vertical integration makes it incredibly difficult for operators to specify high-level network-wide policies using current technologies. Innovation in network management has thus been limited to stop-gap techniques and measures, such as tools that analyze low-level configuration to detect errors or otherwise respond to network events. Proprietary software and closed development in network devices by a handful of vendors make it extremely difficult to introduce and deploy new protocols. Incremental "updates" to configuration methods and commands are generally dictated unilaterally by vendors. Meanwhile, operators' requirements for more functionality and increasingly complex network policies continue to expand.

0163-6804/13/$25.00 © 2013 IEEE

Software defined networking (SDN) is a paradigm where a central software program, called a controller, dictates the overall network behavior. In SDN, network devices become simple packet forwarding devices (data plane), while the "brain" or control logic is implemented in the controller (control plane). This paradigm shift brings several benefits compared to legacy methods. First, it is much easier to introduce new ideas in the network through a software program, as it is easier to change and manipulate than using a fixed set of commands in proprietary network devices. Second, SDN introduces the benefits of a centralized approach to network configuration, opposed to distributed management: operators do not have to configure all network devices individually to make changes in network behavior, but instead make network-wide traffic forwarding decisions in a logically single location, the controller, with global knowledge of the network state.

In this article, we explore how SDN can provide better mechanisms for common network management and configuration tasks across a variety of different types of networks. While many prior studies have explored the potential benefits of applying SDN in computer networks to facilitate the evolution of network technologies(e.g., RCP [5], 4D [6], and Ethane [2]), there has been little study of how SDN might make various tasks associated with managing and operating a network easier.

To allow operators to express and implement reactive high-level policies in an easier manner, we have designed and implemented Procera, an event-driven network control framework based on SDN paradigm. Our policy language and accompanying control framework, Procera, is based on functional reactive programming (FRP). Procera allows operators to express high-level policies with this language, and translates such polices into a set of forwarding rules, which are used to enforce the policy on the underlying network infrastructure, using OpenFlow [10]. We have used Procerato reimplement the existing network policy in the Georgia Tech campus network, which uses complicated VLAN technology and many middleboxes to enforce the campus policy. In combination with the BISmark suite [11], we have implemented a home network management system as well, which does not exist or extremely hard to implement with state-of-the-art legacy configuration methods. Our deployment demonstrates that Procera and SDN can greatly reduce the workload of network configuration and management, and introduce additional functionalities to the network easily.

SDN and OpenFlow

Software defined networking has roots in previous network control systems such as RCP [5], 4D [6], and Ethane [2]. Recent work has introduced the notion of *southbound* and *northbound* interfaces. The southbound interface refers to the interface and protocol between programmable switches (SDN-capable switches) and the software controller. The northbound interface determines how to express operational tasks and network policies, and also how to translate them into a form the controller can understand. In Fig. 1, the protocol between the controller and programmable switch layer is referred to as southbound; northbound refers to the upper part of the controller, including the policy layer.

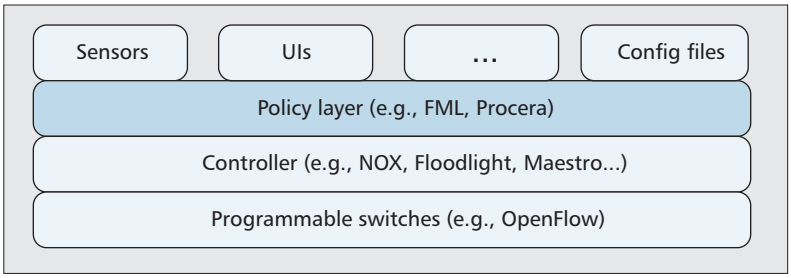

Figure 1. *Layers in software defined networking.*

OpenFlow [10] is one of the most common southbound SDN interfaces. Many vendors, including HP, NEC, NetGear, and IBM, produce OpenFlow-capable network switches available in the market. The Open Networking Foundation (ONF) is responsible for standardizing the OpenFlow protocol. There are a variety of OpenFlow controllers, for example, NOX [7], Floodlight, and Maestro [1]. NOX is a framework that allows developers to program their software program with C++ or Python, using a set of application programming interfaces (APIs) to interact with OpenFlow-capable switches, while Floodlight is a Java-based controller. Maestro focuses on achieving better performance and scalability in a centralized controller using multithreading.

Although there has been much study and industrial effort in defining, polishing, and implementing the southbound part of SDN protocols, there has been relatively little attention on northbound interfaces and protocols. Procera is one effort to define a northbound interface that provides the ability to specify and implement reactive policies.

Procera

Procera is a network control framework that helps operators express event-driven network policies that react to various types of events using a high-level functional programming language. Procera effectively serves as a glue between high-level event-driven network policies and low-level network configuration.

To express event-driven network policies, Procera offers a set of *control domains* that operators can use to set certain conditions and assign appropriate packet forwarding actions corresponding to each condition. Additional control domains can help operators implement flexible, reactive network policies. Operators can also combine control domains to implement rich network policies, instead of relying on time or event-triggered scripts, which are error-prone. The set of control domains Procera supports are summarized in Table 1. We do not claim that the current set of control domains is complete, but it is sufficient to support a range of network

Control domains	Examples
Time	Peak traffic hours, academic semester start date
Data usage	Amount of data usage, traffic rate
Status	Identity of device/user, distinct policy group, authentication status
Flow	Ingress port, ether src/dst, ether type, vlan id, vlan priority, IP src/dst, IP protocol, IP ToS bits, port number src/dst

Table 1. *Control domains in Procera, along with examples of how a higher-level policy can use them.*

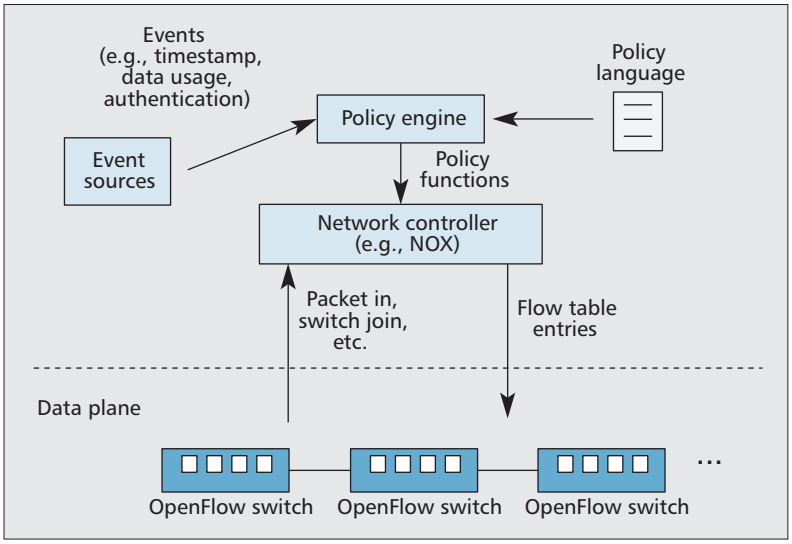

Figure 2. *Procera architecture.*

policies in different types of network environments that are difficult to implement in conventional configuration languages.

Time: Network operators often need to implement policies where network behavior depends on the date or time of day. For example, a campus network operator may want to manage traffic differently in semester breaks when traffic loads are lower than they are during the academic year. In a home network, users might want to use the time of day as the basis for parental control.

Data usage: Operators sometimes specify policies whereby the behavior of the network depends on the amount of data usage (download/upload) or data transfer rate over a particular time interval.

Status: An operator may wish to specify privileges for different users or groups of users. Moreover, a user's privilege or status often changes due to various reasons. A device's privilege should change according to the user who is currently using the device.

Flow: Network operators want to specify different network behaviors based on various field values in multiple layers, specified in a packet or flow. A flow is a 12-tuple control domain that already exists in the OpenFlow specification.

Figure 2 shows the Procera architecture. We elaborate on each component in the following subsections.

Event Sources

Event sources are network components or middleboxes that can send dynamic events to the Procera controller. Intrusion detection systems, network bandwidth monitoring systems, and authentication systems are good examples of event sources. Simple Network Management Protocol (SNMP) or even values in `/proc` can be good event sources as well. As long as there is a parser in the *policy engine* component that understands such events, any kind of event can be raised.

We do not define a fixed interface protocol between event sources and the policy engine, and there can be various alternative methods, such as JSON-RPC. Currently, as a proof of concept, event sources in our deployment periodically send files that contain relevant information, such as the bandwidth usage of every end-host device, along with timestamps.

Policy Engine and Language

The policy engine component is responsible for parsing the network policy expressed with a policy language, and also processing various events that come from event sources. Based on the given policy language and asynchronous events, the policy engine refreshes its *policy state*, which defines the network policy to be enforced, and sends the policy functions to the network controller when the policy state changes. Some reactive policies change the policy state simply according to changes in the time of day, without any external event; the policy language supports these types of reactive changes.

The Procera policy language is based on *functional reactive programming* (FRP). It allows operators to specify complex and reactive network policies in a simple and declarative language. The policy is an embedded domain-specific language in Haskell. Due to scope and page limitations, we do not include details on our policy language in this article; more details are in a work paper on Procera [12].

Network Controller

Procera follows the *software defined networking* paradigm, and thus has a controller that makes all traffic forwarding decisions and updates low-level network switch flow-table entries according to this policy. The *network controller* translates the network policy to actual packet forwarding rules. The network controller establishes a connection to each OpenFlow-capable switch

through the OpenFlow protocol [10], and inserts, deletes, or modifies packet forwarding rules in switches through this connection. The network controller also reacts to packet-in events and switch-join events that come from switches. For packet-in events, the network controller will install relevant forwarding rules in the switch, and for switch-join events, it will establish a new connection with that specific switch. Currently, Procera uses OpenFlow specification version 1.0.0.

Campus Network Deployment

We describe the deployment of Procera in a campus network. Campus networks are dynamic environments with many events occurring across the network. Network policies for campus and enterprise networks are very complex and thus error-prone, which makes them a good subject for deploying Procera.

The Georgia Tech campus network requires every unregistered end-host device to undergo an authentication process via an authentication web portal. After successful authentication with a username and password, the device is scanned for possible vulnerabilities. If none are found, the device is finally granted access to the internal network and the Internet. This simplified version of the actual network policy still involves a complex mechanism that requires input from multiple external tools. In particular, the Georgia Tech campus network relies on virtual LAN (VLAN) technology, where unregistered and registered devices are separated by different VLAN domains. Based on the authentication and scanning results, devices are moved back and forth from two different VLAN domains, and network switches deployed in the network have to constantly download the up-to-date VLAN map from the central VLAN management server (VMPS) to perform correct forwarding behavior.

Implementing such complex and reactive network policy with static tools like firewall rules and VLAN technology requires network operators to independently configure multiple different components, including middleboxes, management servers, and numerous ad hoc scripts. Procera significantly simplifies the expression of these types of policies.

Policy

Figure 3 shows the Georgia Tech campus network policy in terms of a state machine model. The policy can be expressed elegantly with events and transitions among different states. User devices in *unauthenticated* state cannot access the network. Successful authentication with credentials (username and password) moves a device to *scanning* state, where only traffic between the vulnerability scanner is allowed. After no known vulnerabilities are found, a device can transition to the authentication state where the device is finally granted full access to the network. Any infection event from an intrusion detection system can move the device state to *limited*, where access to the network and Internet access are blocked. After five hours of inactivity, the user is required to authenticate again.

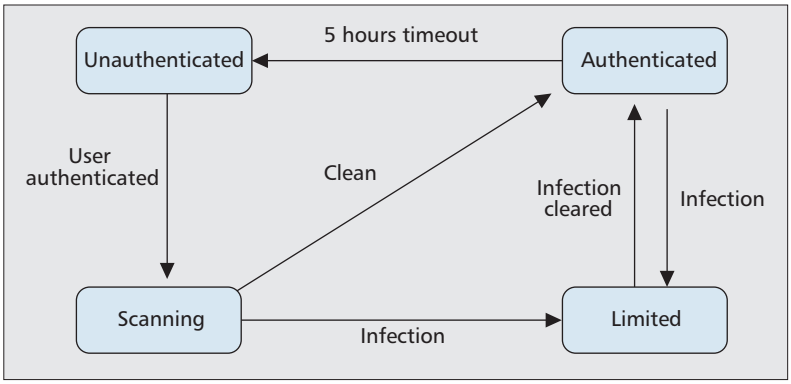

Figure 3. *Transitions and events in campus network.*

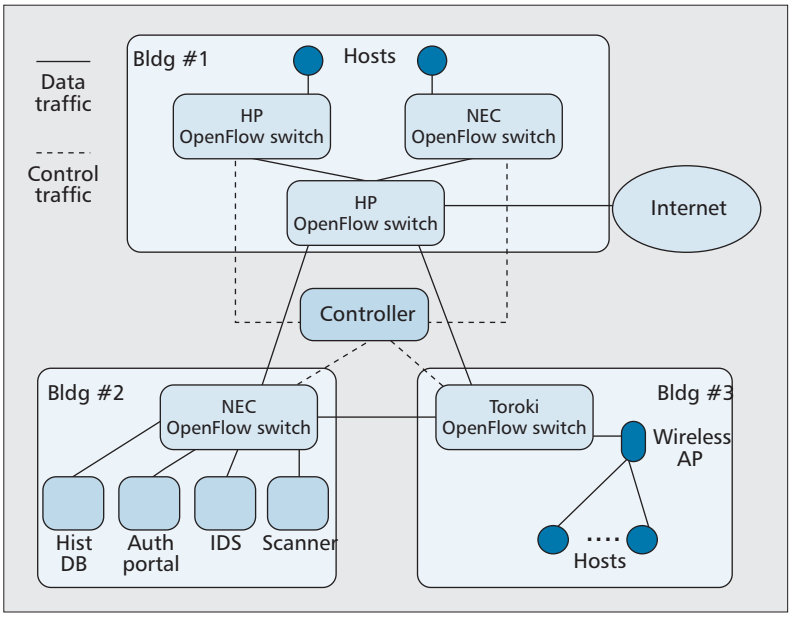

Figure 4. *Campus network deployment.*

Deployment Status

Our campus deployment spans three buildings in the Georgia Tech campus, as shown in Fig. 4. For packet forwarding, we use five OpenFlow-capable network switches from HP, NEC, and Toroki. There are two wireless access points deployed in building 3, through which end-host devices can connect to through a broadcasted SSID. The authentication web portal, intrusion detection system, and scanner, which are event sources, are located in the data closet in building 2.

Home Network Deployments

We describe the deployment of Procera in home networks, and how Procera makes it easier to express various types of policies.

Improving Visibility: BISMark

One of the problems about home networks is that they offer only limited visibility into home broadband performance and its overall status. Measurements performed by individual users with browser-based tools like `speedtest.net` provide limited one-time measurement results,

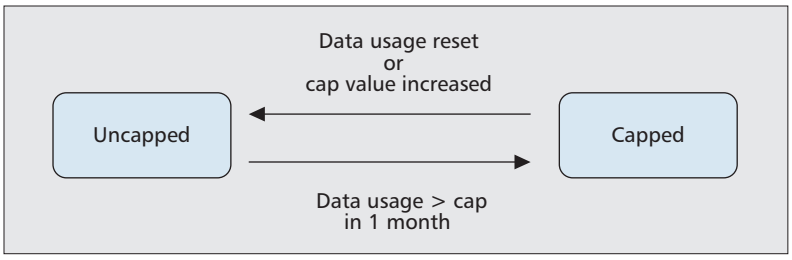

Figure 5. *States and transitions for a simple example of usage cap management in home networks.*

which are likely influenced by many different factors, such as browser type or host computer condition. Access Internet service providers (ISPs) often want to continuously monitor the status of home networks, and ensure that customers receive their promised service. Content providers may desire to know how their traffic engineering decisions influence the home user experience.

BISmark is a collection of home gateways installed in households, a centralized management and data collection server, and multiple measurement servers deployed around the world. The home gateway performs various types of active and passive measurements, which are collected in the centralized management and data collection server for further analysis. As of November 2012, there were around 270 active BISmark gateways deployed around the world. Periodic active and passive measurement results can be used to validate (or invalidate) certain expectations of home networks, and also reveal interesting findings in our Internet [11].

Improving Control: SDN

Due to the inflexible closed software installed in today's common home gateways, it is extremely challenging to introduce new features into the home network. SDN makes it much easier to introduce new functions in these environments.

SDN can introduce new functions for home network traffic management. It is possible to combine BISmark's measurement data and Procera to build a management system that reacts to various conditions of the home network. For example, one possible direction is to perform reactive traffic shaping at the home gateway based on performance measurement results of the home network. Another example is proactively prefetching and caching content from the Internet into the home gateway, even before the last mile. Following the SDN paradigm, enabling a central controller to make various kinds of traffic engineering decisions and pushing rules to home gateways to enforce such policy greatly increases the flexibility of home network management.

Case Study in a Home Network: uCap

We demonstrate how an OpenFlow-enabled home gateway, combined with the BISmark suite, can enable home users to monitor and manage end-host devices' data usage in their home network.

ISPs are starting to enforce monthly bandwidth caps in home networks that limit monthly data usage in the home [3]. Unfortunately, home users currently lack resources to manage or even monitor their daily bandwidth usage. Several ISPs provide a simple web interface where users can monitor their bandwidth usage at the household level, but not at the level of individual devices. Home Internet users need a system and interface where they can not only monitor their daily usage at a device level, but also manage their bandwidth usage by allocating data usage capacity, or caps, to each device. Devices that reach their monthly usage cap can be disabled if the users prefer.

Procera allows users to implement these types of reactive network policies without complex low-level network configuration and ad hoc scripts, which traditional methods would require. The SDN paradigm works well for building such a system, as the logic can be implemented in the back-end server, while routers in home networks can act as simple packet forwarding devices. Through the uCap project [9], we have implemented and deployed such a system in 10 households with OpenFlow-capable wireless home gateways, which communicates with an SDN controller at Georgia Tech.

Policy

Figure 5 shows the basic network policy in the home network deployment. When a device is *uncapped*, a user can access the Internet normally; it is blocked when *capped*. The transition from uncapped to capped occurs when the device's data usage exceeds the monthly cap value, which is set by the home user. The reverse transition is triggered when the cap value is increased or data usage of devices are reset due to the end of a billing cycle. *Event sources* are the wireless routers in households, which send data usage reports back to the back-end server every 5 s. Procera automatically detects any capped or uncapped devices and installs relevant forwarding rules in the wireless router.

Deployment Status

We use NetGear WNDR 3700v2 and 3800 wireless routers as OpenFlow-capable forwarding devices in homes. Home users use the router as a wireless access point in the home and observe no particular difference from any normal wireless access point. Internally, the wireless router runs a customized firmware based on OpenWrt that implements OpenFlow protocol version 1.0.0. As of September 2012, 24 people and 137 devices in 10 households were using the SDN-based system. The total traffic usage of 10 households over three months has reached approximately 1140 Gbytes.

Future Work

Our deployments in campus and home networks demonstrate the feasibility of Procera, but more evaluation on performance and scalability is required. As Procera is based on the *software defined networking* paradigm, it also suffers from the inherent delay caused by the interaction of the control plane and the data plane: packet for-

warding has to wait until the control logic decides on and installs a relevant forwarding rule in the data plane. Recent studies such as DIFANE [13] and DevoFlow [4] focus on resolving performance and scalability issues in the SDN realm through various mechanisms and algorithms. The ongoing research in SDN mechanisms and protocols should help mitigate some of these problems

Currently, Procera has four control domains — time, data usage, authentication status, and traffic flow — that network operators can use to express and implement high-level reactive network policies. This list of domains, however, is far from complete, and there are other possible control domains that can greatly benefit operators when expressing high-level network policies. It is not hard to extend Procera to support more control domains; definition and implementation of how a new event will arrive at Procera, and along with what kind of information would make it possible to support more control domains. This remains as future work.

Procera only supports *allow* and *drop* of packets as possible actions due to a limitation of the OpenFlow 1.0.0 specification. As a result, supporting a richer set of actions (e.g., throughput limit, load balancing, or quality of service) is difficult. Although recent OpenFlow versions have begun to allow basic support for such actions, further improvements to the specification will enable a wider set of packet forwarding actions.

Conclusion

Configuring computer networks is becoming increasingly vexing as network operators must perform increasingly sophisticated network management tasks. Two of the main reasons that networks remain difficult to manage are:

- Continually changing network state
- Low-level per-device network configuration

The state-of-the-art methods for network configuration do not allow network operators to configure network policies that automatically react to low-level networking events, and they do not allow operators to express network policies in terms of high-level intent. Software defined networking introduces a vehicle for raising the level of abstraction for network configuration, and designing languages and network controllers that automatically react to frequent and continual changes to network state.

To simplify various aspects of network operations and management, we have designed and implemented Procera, an event-driven network control framework based on SDN. Network operators can use four control domains — time, data usage, authentication status, and traffic flow — to implement and enforce a reactive network policy with our high-level configuration language based on *functional reactive programming*. We use the OpenFlow protocol to communicate between the Procera controller and the underlying network switches. We have deployed our system in both a campus network and many home networks; our deployment experience and evaluation show that Procera is feasible, introduce more possibilities for expressing a richer set of network policies, and significantly reduce the complexity of network management in a variety of network settings and for a range of network policies.

We have deployed our system in both a campus network and many home networks; our deployment experience and evaluation shows that Procera is feasible, introduces more possibilities for expressing a richer set of network policies, and significantly reduces the complexity of network management in a variety of network settings and for a range of network policies.

References

[1] Z. Cai, *Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane*, Ph.D. thesis, 2011.
[2] M. Casado *et al.*, "Rethinking Packet Forwarding Hardware," *Proc. 7th ACM SIGCOMM HotNets Wksp.*, Nov. 2008.
[3] M. Chetty *et al.*, "You're Capped: Understanding the Effects of Bandwidth Caps on Broadband Use in the Home," *Proc. 2012 ACM Annual Conf. Human Factors in Computing Systems*, CHI '12, New York, NY, 2012, pp. 3021–30.
[4] A. R. Curtis *et al.*, "Devoflow: Scaling Flow Management for High-Performance Networks," *Proc. ACM SIGCOMM '11*, New York, NY, 2011, pp. 254–65.
[5] N. Feamster *et al.*, "The Case for Separating Routing from Routers," *ACM SIGCOMM Wksp. Future Directions in Network Architecture*, Portland, OR, Sept. 2004.
[6] A. Greenberg *et al.*, "A Clean Slate 4D Approach to Network Control and Management," *ACM Comp. Commun. Rev.*, vol. 35, no. 5, 2005, pp. 41–54.
[7] N. Gude *et al.*, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, no. 3, July 2008, pp. 105–10.
[8] H. Kim *et al.*, "The Evolution of Network Configuration: A Tale of Two Campuses," *Proc. 2011 ACM SIGCOMM Conf. Internet Measurement Conf.*, New York, NY, 2011, pp. 499–514.
[9] H. Kim *et al.*, "Communicating with Caps: Managing Usage Caps in Home Networks," *Proc. ACM SIGCOMM '11*, New York, NY, 2011, pp. 470–71.
[10] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM Comp. Commun. Rev.*, Apr. 2008.
[11] S. Sundaresan *et al.*, "Broadband Internet Performance: A View from the Gateway," *Proc. ACM SIGCOMM*, Toronto, Ontario, Canada, Aug. 2011.
[12] A. Voellmy, H. Kim, and N. Feamster, "Procera: A Language for High-Level Reactive Network Control," *Proc. 1st Wksp. Hot Topics in Software Defined Networks*, New York, NY, 2012, pp. 43–48.
[13] M. Yu *et al.*, "Scalable Flow-Based Networking with DIFANE," *Proc. ACM Sigcomm*, Aug. 2010.

Biographies

Hyojoon Kim (joonk@gatech. edu) is with Georgia Institute of Technology.

Nick Feamster (feamster@cc. gatech. edu) is with Georgia Institute of Technology.