IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 1, Ver. I (Jan – Feb. 2016), PP 30-36
www.iosrjournals.org

Enhancement of Network Administration through Software Defined Networks

[1]Dr.V.V.S.S.S.Balaram, [2]Dr.Chinthagunta Mukundha, [3]Sunil Bhutada
[1]*Professor&HOD,Dept of IT, SNIST,Hyderabad-501301*
[2]*Associate Professor,Dept of IT,SNIST,Hyderabad-501301*
[3]*Associate Professor,Dept of IT,SNIST,Hyderabad-501301*

Abstract: *Now a days organizing the Network is very complex and challenging issue. To control, manage, and to provide a secure communication network, network managers must grapple with low-level vendor-specific configuration to implement complex high-level network policies .Most of the previous proposals to make networks easier to manage, many solutions to network management problems amount to stop-gap solutions because of the complexity of changing the underlying infrastructure. The rigidity of the underlying infrastructure presents few considerations for innovation or improvement, since network devices have generally been closed, proprietary, and vertically integrated.*

A new approach in networking, software defined networking (SDN), suggests separating the data plane and the control plane, making network switches and routers in the data plane simple packet forwarding devices and leaving a logically centralized software program to manage the behavior of the entire network. SDN introduces new possibilities for network management and configuration methods. In this paper, 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: first one enabling frequent changes to network conditions and state, second providing support for network configuration in a highlevel language, and third providing better visibility and control over tasks for performing network analysis and troubleshooting.

Keywords: *Network ,configuration ,infrastructure ,routers , diagnosis*

I. 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 idle boxes 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 adhoc scripts to dynamically reconfigure network devices when events occur. As a result, configuration changes are frequent and unwieldy, leading to frequent misconfigurations . 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. Mean while, operators' requirements for more functionality and increasingly complex network policies continue to expand.

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 , 4D , and Ethane), 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 highlevel 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 .We have used Procera to 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 , 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.

II. Need For A New Network Architecture :

The explosion of mobile devices and content, server virtualization, and advent of cloud services are among the trends driving the networking industry to reexamine traditional network architectures. Many conventional networks are hierarchical, built with tiers of Ethernet switches arranged in a tree structure. This design made sense when client-server computing was dominant, but such a static architecture is ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments. Some of the key computing trends driving the need for a new network paradigm include:

Changing traffic patterns: Within the enterprise data center, traffic patterns have changed significantly. In contrast to client-server applications where the bulk of the communication occurs between one client and one server, today's applications access different databases and servers, creating a flurry of "east-west" machine-to-machine traffic before returning data to the end user device in the classic "north-south" traffic pattern. At the same time, users are changing network traffic patterns as they push for access to corporate content and applications from any type of device (including their own), connecting from anywhere, at any time. Finally, many enterprise data centers managers are contemplating a utility computing model, which might include a private cloud, public cloud, or some mix of both, resulting in additional traffic across the wide area network.

The "consumerization of IT": Users are increasingly employing mobile personal devices such as smartphones, tablets, and notebooks to access the corporate network. IT is under pressure to accommodate these personal devices in a fine-grained manner while protecting corporate data and intellectual property and meeting compliance mandates.

The Rise of Cloud Services: Enterprises have enthusiastically embraced both public and private cloud services, resulting in unprecedented growth of these services. Enterprise business units now want the ability to access applications, infrastructure, and other IT resources on demand and à la carte. To add to the complexity, IT's planning for cloud services must be done in an environment of increased security, compliance, and auditing requirements, along with business reorganizations, consolidations, and mergers that can change assumptions overnight. Providing self-service provisioning, whether in a private or public cloud, requires elastic scaling of computing, storage, and network resources, ideally from a common viewpoint and with a common suite of tools.

"Big data" means more bandwidth: Handling today's "big data" or mega datasets requires massive parallel processing on thousands of servers, all of which need direct connections to each other. The rise of mega datasets is fueling a constant demand for additional network capacity in the data center. Operators of hyperscale data center networks face the daunting task of scaling the network to previously unimaginable size, maintaining any-to-any connectivity without going broke.

III. Limitations Of Current Networking Technologies

Meeting current market requirements is virtually impossible with traditional network architectures. Faced with flat or reduced budgets, enterprise IT departments are trying to squeeze the most from their networks using device-level management tools and manual processes. Carriers face similar challenges as demand for mobility and bandwidth explodes; profits are being eroded by escalating capital equipment costs and flat or declining revenue. Existing network architectures were not designed to meet the requirements of today's users, enterprises, and carriers; rather network designers are constrained by the limitations of current networks, which include:

Complexity that leads to stasis: Networking technology to date has consisted largely of discrete sets of protocols designed to connect hosts reliably over arbitrary distances, link speeds, and topologies. To meet business and technical needs over the last few decades, the industry has evolved networking protocols to deliver higher performance and reliability, broader connectivity, and more stringent security.

Protocols tend to be defined in isolation, however, with each solving a specific problem and without the benefit of any fundamental abstractions. This has resulted in one of the primary limitations of today's networks: complexity. For example, to add or move any device, IT must touch multiple switches, routers, firewalls, Web authentication portals, etc. and update ACLs, VLANs, quality of services (QoS), and other protocol-based mechanisms using device-level management tools. In addition, network topology, vendor switch model, and software version all must be taken into account. Due to this complexity, today's networks are relatively static as IT seeks to minimize the risk of service disruption.

The static nature of networks is in stark contrast to the dynamic nature of today's server environment, where server virtualization has greatly increased the number of hosts requiring network connectivity and fundamentally altered assumptions about the physical location of hosts. Prior to virtualization, applications resided on a single server and primarily exchanged traffic with select clients. Today, applications are distributed across multiple virtual machines (VMs), which exchange traffic flows with each other. VMs migrate to optimize and rebalance server workloads, causing the physical end points of existing flows to change (sometimes rapidly) over time. VM migration challenges many aspects of traditional networking, from addressing schemes and namespaces to the basic notion of a segmented, routing-based design.

In addition to adopting virtualization technologies, many enterprises today operate an IP converged network for voice, data, and video traffic. While existing networks can provide differentiated QoS levels for different applications, the provisioning of those resources is highly manual. IT must configure each vendor's equipment separately, and adjust parameters such as network bandwidth and QoS on a per-session, per-application basis. Because of its static nature, the network cannot dynamically adapt to changing traffic, application, and user demands.

Inconsistent policies: To implement a network-wide policy, IT may have to configure thousands of devices and mechanisms. For example, every time a new virtual machine is brought up, it can take hours, in some cases days, for IT to reconfigure ACLs across the entire network. The complexity of today's networks makes it very difficult for IT to apply a consistent set of access, security, QoS, and other policies to increasingly mobile users, which leaves the enterprise vulnerable to security breaches, non-compliance with regulations, and other negative consequences.

Inability to scale: As demands on the data center rapidly grow, so too must the network grow. However, the network becomes vastly more complex with the addition of hundreds or thousands of network devices that must be configured and managed. IT has also relied on link oversubscription to scale the network, based on predictable traffic patterns; however, in today's virtualized data centers, traffic patterns are incredibly dynamic and therefore unpredictable.Mega-operators, such as Google, Yahoo!, and Facebook, face even more daunting scalability challenges. These service providers employ large-scale parallel processing algorithms and associated datasets across their entire computing pool. As the scope of end-user applications increases (for example, crawling and indexing the entire world wide web to instantly return search results to users), the number of computing elements explodes and data-set exchanges among compute nodes can reach petabytes. These companies need so-called hyperscale networks that can provide high-performance, low-cost connectivity among hundreds of thousands— potentially millions—of physical servers. Such scaling cannot be done with manual configuration.

To stay competitive, carriers must deliver ever-higher value, better-differentiated services to customers. Multi-tenancy further complicates their task, as the network must serve groups of users with different applications and different performance needs. Key operations that appear relatively straightforward, such as steering a customer's traffic flows to provide customized performance control or on-demand delivery, are very complex to implement with existing networks, especially at carrier scale. They require specialized devices at the

network edge, thus increasing capital and operational expenditure as well as time-to-market to introduce new services.

Vendor dependence: Carriers and enterprises seek to deploy new capabilities and services in rapid response to changing business needs or user demands. However, their ability to respond is hindered by vendors' equipment product cycles, which can range to three years or more. Lack of standard, open interfaces limits the ability of network operators to tailor the network to their individual environments.

This mismatch between market requirements and network capabilities has brought the industry to a tipping point. In response, the industry has created the Software-Defined Networking (SDN) architecture and is developing associated standards.

V. Openflow And Sdn

OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). It is the absence of an open interface to the forwarding plane that has led to the characterization of today's networking devices as monolithic, closed, and mainframe-like. No other standard protocol does what OpenFlow does, and a protocol like OpenFlow is needed to move network control out of the networking switches to logically centralized control software.

Software Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable. This migration of control, formerly tightly bound in individual network devices, into accessible computing devices enables the underlying infrastructure to be abstracted for applications and network services, which can treat the network as a logical or virtual entity.

Figure 1 depicts a logical view of the SDN architecture. Network intelligence is (logically) centralized in software-based SDN controllers, which maintain a global view of the network. As a result, the network appears to the applications and policy engines as a single, logical switch. With SDN, enterprises and carriers gain vendor-independent control over the entire network from a single logical point, which greatly simplifies the network design and operation. SDN also greatly simplifies the network devices themselves, since they no longer need to understand and process thousands of protocol standards but merely accept instructions from the SDN controllers.

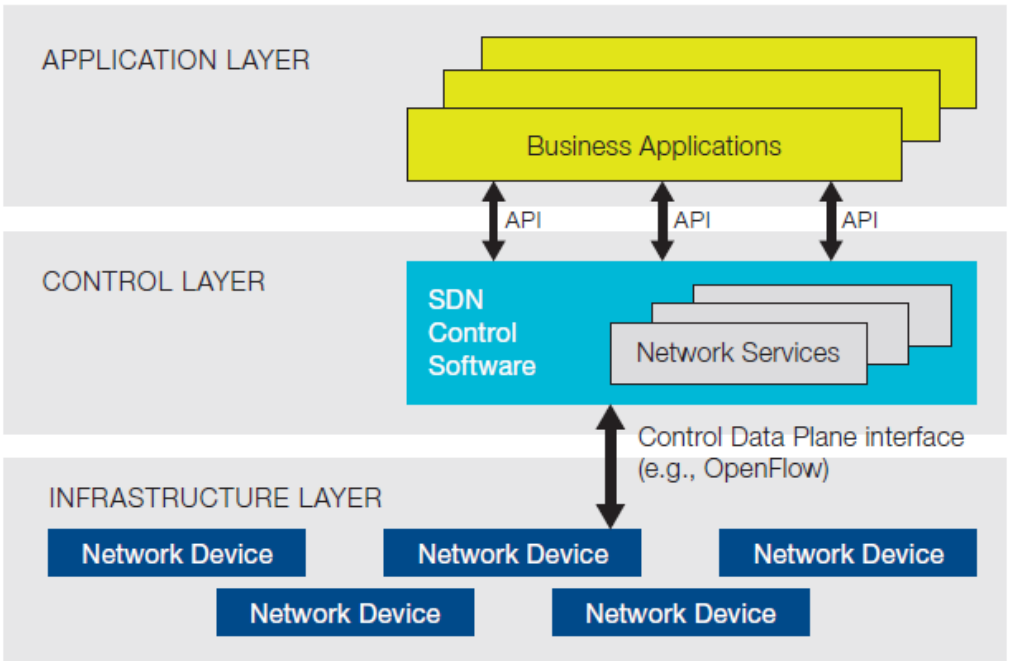

Figure 1 : Layers of Software Defined Networks

Perhaps most importantly, network operators and administrators can programmatically configure this simplified network abstraction rather than having to hand-code tens of thousands of lines of configuration scattered among thousands of devices. In addition, leveraging the SDN controller's centralized intelligence, IT can alter network behavior in real-time and deploy new applications and network services in a matter of hours or

days, rather than the weeks or months needed today. By centralizing network state in the control layer, SDN gives network managers the flexibility to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs. Moreover, they can write these programs themselves and not wait for features to be embedded in vendors' proprietary and closed software environments in the middle of the network.

In addition to abstracting the network, SDN architectures support a set of APIs that make it possible to implement common network services, including routing, multicast, security, access control, bandwidth management, traffic engineering, quality of service, processor and storage optimization, energy usage, and all forms of policy management, custom tailored to meet business objectives. For example, an SDN architecture makes it easy to define and enforce consistent policies across both wired and wireless connections on a campus.

Likewise, SDN makes it possible to manage the entire network through intelligent orchestration and provisioning systems. The Open Networking Foundation is studying open APIs to promote multi-vendor management, which opens the door for on-demand resource allocation, self-service provisioning, truly virtualized networking, and secure cloud services. Thus, with open APIs between the SDN control and applications layers, business applications can operate on an abstraction of the network, leveraging network services and capabilities without being tied to the details of their implementation. SDN makes the network not so much "application-aware" as "application-customized" and applications not so much "network-aware" as "network-capability-aware". As a result, computing, storage, and network resources can be optimized.

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

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.*

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.

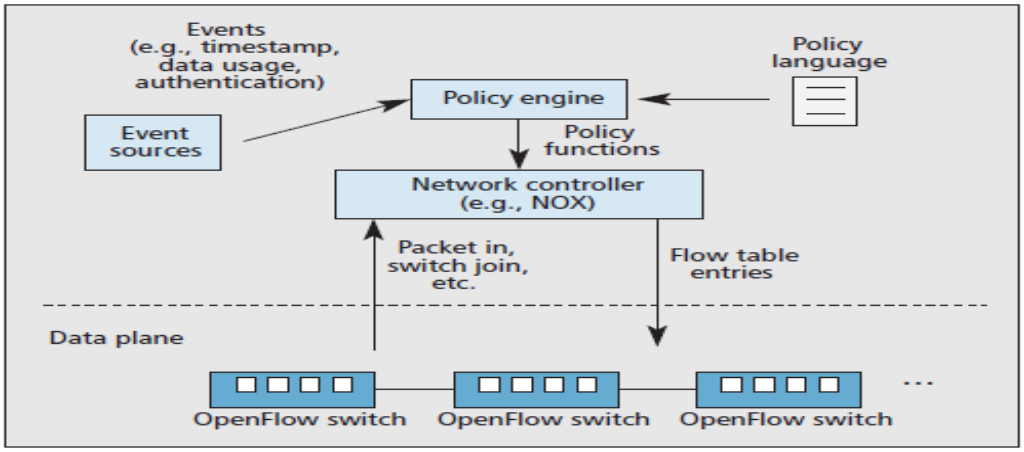

Figure 2 : Procera Architecture

VI. 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.

VII. 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 supportsthese 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.

VIII. 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 , 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.

IX. 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 ofthe 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 and DevoFlow 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.

X. Conclusion

Trends such as user mobility, server virtualization, IT-as-a-Service, and the need rapidly to respond to changing business conditions place significant demands on the network—demands that today's conventional network architectures can't handle. Software-Defined Networking provides a new, dynamic network architecture that transforms traditional network backbones into rich service-delivery platforms. By decoupling the network control and data planes, OpenFlow-based SDN architecture abstracts the underlying infrastructure from the applications that use it, allowing the network to become as programmable and manageable at scale as the computer infrastructure that it increasingly resembles. An SDN approach fosters network virtualization, enabling IT staff to manage their servers, applications, storage, and networks with a common approach and tool set. Whether in a carrier environment or enterprise data center and campus, SDN adoption can improve network manageability, scalability, and agility.

References

[1] Z. Cai, Maestro: Achieving Scalability and Coordination inCentralized 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 theEffects 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 Languagefor 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 withDIFANE," Proc. ACM Sigcomm, Aug. 2010.