Participatory Networking

Andrew D. Ferguson, Arjun Guha, Jordan Place, Rodrigo Fonseca and Shriram Krishnamurthi
Brown University

Abstract

Software Defined Networks, which provide a programmable, logically centralized abstraction of network control, offer an escape from the current state of enterprise and datacenter network configuration, plagued by brittle, static solutions involving manual setting of myriad devices. But if SDNs provide an operating system for the network, we are missing the analog to system calls – an API for end-users and their applications to take part in network configuration. In response, we propose *participatory networking*, a new paradigm for network configuration in which users submit requests or hints for current and future network properties such as quality of service, access control, and path selection. We describe the initial design and implementation of a participatory networking system, PANE, and its solutions to the challenges of resource arbitration and privilege delegation.

1 Introduction

The configuration of today's datacenter and enterprise networks is not for the faint of heart. Network administrators develop brittle, static solutions to implement desired policies, which exist as an interwoven mix of VLANs, firewalls, router configurations, MPLS tunnels, and more. This complicates delegation of control, sacrifices flexibility, and increases the time to fix errors and vulnerabilities to human scale.

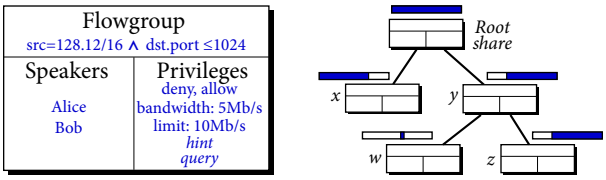
The recent development of Software Defined Networks (SDNs) offers a platform for simplification [8, 14]. SDNs separate the logic that controls the network from its physical devices, allowing configuration *programs* to operate on a high-level, global, and consistent view of the network. This change has brought significant advances to datacenter and enterprise networks, such as high-level specification of access control [16] and QoS [12], safe experimentation [19], in-network load-balancing [23], and seamless VM migration [5].

While SDNs bring power and flexibility to configuring networks, we argue that the current stack is incomplete. If SDNs provide an operating system for the network [9], we are missing a mechanism analogous to system calls, with which less privileged users can request services from the network and influence its operation.

We propose a new paradigm for network configuration, which we term *participatory networking*. In this paper we present the initial design and implementation of a participatory networking system, PANE, which forms the "user-level" interface for the network control-plane, and serves as the next layer for the current SDN stack. Here, "user" encompasses both operators and end-users, including applications and devices acting on their behalf. Our vision is that operators set baseline policies that guarantee fairness and security, while devices, applications, and end-users request current and future service qualities, gain visibility into network properties, and provide hints about future demands to the network. One of the key characteristics of PANE is that it considers the temporal dimension of requests, and can reason beyond current conditions for resource allocation and configuration.

For example, a conferencing application may request bandwidth for a video call, but learn via the network that while only a guaranteed audio call is available now, it can reserve a video call in one hour. An intrusion detection script on a user machine may request that the *network* filter traffic from a specific source. An RPC service may inform the network that its flows will be short, but with a deadline, which could enable shortest-deadline-first scheduling at the switches [24]. Or, a MapReduce-style application may request maximally disjoint paths to improve performance of its shuffle phase. While not all of these are new, PANE provides a *unified* framework for dynamic user involvement in network-wide configuration.

PANE assumes, and is made deployable by, a logically centralized control-plane, which SDNs provide. Our initial focus is on networks in a single administrative domain, such as an enterprise, campus, or datacenter. PANE does not provide any functionality not already present in the underlying network: it exposes it to a (much) broader set of users, defining the necessary policies for sharing and arbitration. For example, PANE does not invent QoS mechanisms, but can use QoS extensions to OpenFlow [12], to implement bandwidth and latency guarantees for flows. Unlike in distributed reservation protocols such as RSVP [3], the PANE controller has a global view of the network, and does not rely on independent decisions by network elements. PANE's main contribution is to use this global view of the network and requests to arbitrate



(a) (b)

Figure 1: (a) A share in PANE. (b) A share hierarchy. The rectangle above each share represents a flowgroup according to one dimension (*e.g.*, source IP). Sub-shares are defined on a subset of their parent's flowgroup, and may not have more permissive privileges than their parent.

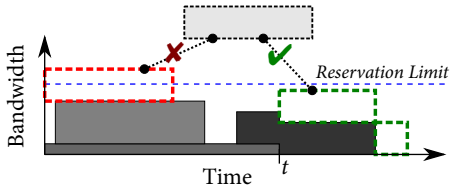


Figure 2: Example user request for reserved bandwidth; PANE determines that it cannot be fulfilled until time t.

among users, much like a traditional OS. Of course, for this arbitration to work, we need well-defined semantics for the delegation of privileges in the network.

Our approach to network configuration is backwards-compatible with existing networked applications – users submit requests only to receive *predictable* behavior from the network. Unmodified applications continue to receive the best-effort performance of existing packet networks.

PANE has three main components: first, the semantics of privilege delegation required to reconcile requests and the network's constraints (§2 and §4.1); second, a protocol and API for operators and users (or their applications) to interact with the network (§4.2); and third, a network controller that implements the protocol according to the semantics and installs policies in the network (§5). We have implemented a prototype controller that runs using Nettle [22] and OpenFlow[14], and allows immediate and future bandwidth reservations and installation of traffic permission rules, which we evaluate through example applications in §5. PANE is a work in progress, and builds upon previous work on resource sharing, network QoS, and software-defined networking, which we discuss in §6.

2 The Essence of PANE

PANE provides a mechanism to safely delegate privileges that affect shared network resources. The principals in PANE are *speakers*, who are end-users, or applications

Speaker	A network user authorized to issue messages.
Flow	Set of packets with shared characteristics (*e.g.*, transport protocol or dest. MAC address).
Flowgroup	A set of flows. Can be specified by any constraints on flow attributes, such as 'all flows directed to TCP port 80', or 'all flows from Alice'.
Privilege	The right to issue a message on a flowgroup.
Share	Associated flowgroup, speakers, privileges.

Table 1: Main concepts in PANE

and devices running on their behalf.[1] Speakers interact with the network using three types of messages: *requests*, *hints*, and *queries*. The PANE controller checks the validity of messages, as described below, and may apply changes to the network and return a response. Requests are for resources (*e.g.*, bandwidth or access control), an action to be taken by the controller, and an immediate response. Queries imply no action, but receive a response. Hints inform the network about current or future traffic characteristics and never have a response; PANE may choose to use the hint to improve service to users. Messages always refer to a *flowgroup*, a subset of all possible flows on the network. Finally, the definition and delegation of *privileges* – which *speakers* can issue which *messages* on which *flowgroups* – forms the essence of PANE.

PANE models the delegation of privileges using a hierarchy of *shares*. A share is a combination of a flowgroup, a set of speakers, and a set of privileges (Figure 1(a)); the share's speakers can issue messages allowed by the privileges on subsets of the flowgroup. An implicit privilege of all shares is delegation: speakers can always create sub-shares (Figure 1(b)) or add new speakers. Sub-shares may have an arbitrary set of speakers, must refer to a subset of the parent share's flowgroup, and may not have more permissive privileges than their parent. This codifies the intuition that "you can't give more privileges than you have, but you can give them to anyone." For example, if a share's flows may have a maximum guaranteed bandwidth of B, a sub-share cannot provide guaranteed bandwidth of $B' > B$. We describe the analogous relation for additional privileges in §4.1.

The share hierarchy does not itself change the state of the network; the network is changed by speakers' requests and, optionally, hints, which are issued on a specified flowgroup and share. PANE accepts a message if it passes a privilege check, if the referenced flowgroup is a subset of the share's flowgroup, and, for requests, if the request can coexist with previously accepted requests. By default, requests take effect immediately and do not expire, but can optionally specify start and end times. PANE verifies the request's feasibility into the future, between the provided or implicit start and end times. Verifying if

[1] Users in PANE are authenticated using mechanisms such as 802.1x.

a request can be granted may require walking the share hierarchy, depending on the resource. The design of subshares allows resources to be oversubscribed; overallocation is prevented when requests are granted, and not when shares are created.

Figure 2 shows a simple example in which a speaker has requested an immediate bandwidth reservation. PANE determines that granting the request will exceed the share's available bandwidth, and informs the user that the request cannot be granted until time t. In response, the speaker sends a new request for a reservation which starts at t; PANE accepts the request and later implements it.

With these definitions in place (summarized in Table 1), we now consider several example scenarios in which PANE benefits a network's users. We defer the discussion of further design details, including the mechanism for resolving conflicting requests, until §4.1.

3 Motivating Examples

We illustrate potential benefits of PANE with four motivating scenarios in which there are significant gains when users or applications interact directly with the network.[2]

Signal Bars for Video Conferencing Consider a user trying to establish a video conference when the network is operating near capacity. Today, the user will place the call blindly, and become frustrated by its poor quality. Alternatively, in a network managed by PANE, the conferencing application can issue a request for the required bandwidth. The network, knowing that a large file transfer will end in 20 minutes, responds to the application request with the schedule of bandwidth guarantees. With this information, the application can reserve enough bandwidth now for a good quality audio call, or, in 20 minutes, a good quality video call. By setting expectations and offering alternatives, the conferencing application provides a less frustrating experience for the user.

Network-assisted Firewall When a host is attacked with undesired traffic, automated scripts can install local firewall rules to drop such packets.[3] The user can also contact the network administrator to request malicious traffic be filtered earlier in the network. While the latter option has technical advantages, it is often too inconvenient in practice. With PANE and appropriate shares, the same script can programmatically prevent the attacking host from reaching its target. This approach shares many of the advantages of ident++'s delegated firewall rules [15], to which we compare in §6.

Hints for Traffic Prioritization The third scenario concerns the prioritization or scheduling of flows with help from users or applications. For example, by augmenting the Linux packet scheduler to support priorities provided by Apache, the mean response time of short flows can be provably reduced without adversely affecting longer flows [10]. With PANE, Apache could push these priorities beyond the local node to all programmable network elements by specifying the relevant flowgroups and either requesting guaranteed treatment to them, or providing PANE with priority hints. Similarly, for deadline-driven workloads such as those considered by D^3 [24], PANE provides the right platform for applications to pass this information to the network, which can use it to install QoS rules along the flow's path. In this case, the hints about the relevant flowgroups could contain the deadline and size of each flow. We expand upon this example in §6.

Finally, the direct use of application information obviates the need for heuristics to identify traffic which may look similar from the perspective of the network, such as long-lived HTTP flows for either streaming video or networked backup. Using PANE, a user can request that the first have less jitter and lower latency than the second.

Network Path Configuration Hybrid optical-electrical networks in datacenters have shown performance benefits for applications requiring large transfers, such as virtual machine migration, MapReduce-style computation, and HPC workloads. However, proposers of two such designs acknowledge that their efficient use "relies on precise and detailed traffic analysis" [2]. Hedera [1], a scheme for dynamic flow allocation in datacenter topologies with multi-rooted trees, also requires estimates of the traffic demand between pairs of hosts. PANE offers an appealing alternative to reactive traffic estimation: application-provided hints about future traffic patterns would allow the network controller to allocate optical links or bandwidth to flows without resorting to heuristics.

While these examples highlight potential benefits of our system, they also raise many design and implementation challenges. We currently address some of these in PANE, but leave others, such as the hints of the last two scenarios, as future work.

4 Design

PANE's design has two main parts: the semantics of the share hierarchy, privileges, and messages, and the PANE protocol. We deepen the discussion from §2, and provide a sketch of the PANE protocol by way of examples.

4.1 Privileges and Requests

The share hierarchy sets the (static) context for the evaluation of speaker messages. The root of the share hierarchy is the RootShare, which contains all flows, has all

[2] Throughout the paper, when we say interact with the *network*, we mean with the logically centralized network control plane.

[3] For example, SSHGuard: http://www.sshguard.net

Privilege	Resource	Constraints	Sub-shares	Composition
Allow Deny	Access Control	boolean	same or none	Parent: override Sibling: Deny overrides Allow
Reserve	Bandwidth	Token Bucket	Capacity: ≤; Fill Rate: ≤; Min Drain: ≥; Max Drain: ≤	Draw from all TB up to root. Simulate forward in time.
Limit	Bandwidth	Maximum	≤	≤ parent

Table 2: Share privileges for types of requests implemented in our PANE prototype.

privileges, and for which the root user is the only speaker. Speakers of a share can create any number of sub-shares and attribute both speakers and privileges to the sub-share. Flowgroups are specified with constraints on the flow attributes. We currently support the same attributes as Flow-based Management Language [11], plus transport ports. Using the labeling of Figure 1(b), flowgroups of sub-shares can be proper ($w \subset y$) or improper ($z \equiv y$) subsets of their parents' flowgroups, and flowgroups of sibling sub-shares (shares x and y) can overlap. The share hierarchy, by virtue of subdividing the space of possible flows, provides a natural context for the evaluation of all three message types: requests, hints, and queries. In this paper, we focus on the definition and implementation of requests, leaving the other types for future work.

Our prototype implements four request types, summarized in Table 2. A speaker can use *allow* and *deny* to specify access control rules, *reserve* to obtain a minimum bandwidth guarantee, and *limit* to rate-limit a flowgroup's bandwidth. We specify bandwidth reservation privileges as a modified token bucket: it has the usual attributes of fill rate F, capacity C, and maximum drain rate M, and an additional minimum drain rate m. This lower bound prevents reservations with very low drain rates that could last indefinitely. A simple reservation with maximum bandwidth B is a special case with $F = M = B; C = m = 0$.

The privileges of a sub-share cannot be less restrictive than those of its parent share (cf. Table 2, 'Sub-shares' column). For both *allow* and *deny*, the sub-share can maintain the parent share's privilege, if present, or revoke it. The token bucket of a sub-share has to "fit inside" the parent's token bucket. For *limit*, sub-shares can only specify limits that are smaller than or equal to the parent's.

PANE maintains the known state of the network – the set of active and future granted requests – and evaluates whether new requests can co-exist with them, and if so, their effect. The last column of Table 2 describes the composition of privileges along the share hierarchy.

For each request, the speaker can specify an evaluation mode, either *strict* or *partial*, which are provided for atomicity and convenience. In strict mode, PANE rejects a request if it conflicts in any way with the network state. For example, if a user wants to allow connections to TCP ports 1000-2000, but there exists a request in a sub-share that denies port 1024, PANE rejects the request, explaining why. In partial mode, PANE relaxes the request so that it does not conflict, and informs the user of the change. In the same example, PANE would inform the user that it has allowed ports 1000-1023, and 1025-2000.

We believe that requests for other types of resources, such as latency guarantees or path properties such as *waypoint* and *avoid*, will fit in the same framework, provided we can define what "less restrictive" means for each one. We leave this investigation, however, to future work.

4.2 The PANE Protocol

Speakers talk to the PANE controller using a simple protocol sketched here, beginning with the requests necessary for an end-user, Alice, to reserve guaranteed bandwidth for her flows. First, the network administrator must create a share for her flows which carries such a privilege:

```
NewShare aliceBW for (user=Alice)
  [reserve <= 10Mb] on rootShare.
```

This share's flowgroup contains all flows for which Alice is the sender (`user=Alice`) and allows the share's speakers to reserve up to 10Mbps of guaranteed bandwidth, but does not authorize *allow* or *deny* requests. Next, the administrator makes Alice a speaker of this share:

```
Grant aliceBW to Alice.
```

Finally, Alice can request that her web traffic receive guaranteed bandwidth of 5Mbps for ten minutes, starting twenty minutes in the future:

```
reserve(user=Alice,dstPort=80) = 5Mb
  on aliceBW from +20min to +30min.
```

While `aliceBW` permits Alice to reserve bandwidth indefinitely, a token bucket can be applied to limit Alice's reservations; for brevity, we omit syntax for this example.

The network administrator can also create shares to delegate access control. The following PANE request creates a share for Alice, who is using the IP address 10.0.0.2, which allows her to deny traffic to her computer, but which cannot be used for other requests:

```
NewShare aliceAC for (dstHost=10.0.0.2)
  [deny = True] on rootShare.
Grant aliceAC to Alice.
```

Using this share, Alice can, for example, block traffic from Eve's host (10.0.0.3) for the next five minutes:

```
deny(dstHost=10.0.0.2, srcHost=10.0.0.3)
  on aliceAC from now to +5min.
```

If Alice attempts to block traffic to Bob's computer (10.0.0.4) with this command:

```
deny(dstHost=10.0.0.4, srcHost=10.0.0.3)
  on aliceAC.
```

her request will be rejected because its flowgroup is not a subset of the flowgroup in `aliceAC`. If Alice attempts to block all traffic from Eve by not including the `dstHost` restriction, her request will also be rejected.

We have also implemented, but omit for space reasons, commands to query a share's schedule of reserved bandwidth, determine which shares and requests affect which flows, and manage users and their privileges.

5 Prototype

We have developed an initial Haskell implementation of PANE for OpenFlow networks using Nettle [22]. It has been tested using the Mininet platform for emulating SDNs and the Open vSwitch implementation of an OpenFlow switch. Although we chose OpenFlow as our substrate for implementing PANE, its design does not depend on OpenFlow. PANE could be implemented using other mechanisms to control the network, such as 4D [8], MPLS, or a collection of middleboxes. With our prototype we have implemented these preliminary case studies:

Network-assisted firewall SSHGuard is a popular tool to detect brute-force attacks via log monitoring and install local firewall rules (*e.g.*, via `iptables`) in response. We modified SSHGuard to use PANE as a firewall backend to block nefarious traffic entering the network. In particular, this means such traffic no longer traverses the targeted host's access link.

For example, if Alice is running SSHGuard on her host and it detects a Linux syslog entry such as:

```
sshd[2197]: Invalid user Eve from 10.0.0.3
```

SSHGuard will block Eve's traffic for the next five minutes using the *deny* command presented in §4.2. In our prototype, this PANE request is then installed as a packet forwarding rule on the OpenFlow switches, which drop packets to Alice's host coming from Eve's.

Ekiga Ekiga is an open source video conferencing application. We modified Ekiga to ask the user for the anticipated duration of video calls, and use a *reserve* command to request guaranteed bandwidth from the network for the appropriate time. If such a reservation is not available, Ekiga retrieves the schedule of available bandwidth from PANE and calculates the earliest time at which a video call or, alternatively, an audio call, can be made with guaranteed quality. In our prototype, a successful request is installed as an OpenFlow rule which sends Ekiga's packets through a preconfigured queue on the switch.

6 Related Work

Quality of Service and Resource Management Providing a predictable network experience is not a new goal, and there is a vast body of literature on this topic. PANE relies heavily on existing mechanisms, such as reservations and prioritized queue management [12, 20], while adding user-level management and resource arbitration. PANE also goes beyond QoS, integrating hints and guarantees about access control and path selection. By focusing on a single administrative domain, PANE sidesteps the deployment difficulties of Internet-wide QoS proposals such as IntServ [4] and DiffServ [3].

Cinder [18] uses a hierarchy of *taps* to provide isolation, delegation, and division of the right to consume a mobile device's energy, similar to PANE's hierarchy of token buckets for managing bandwidth reservations.

Software Defined Networking Recent developments in making SDN practical (*e.g.*,[9, 14, 22]) greatly improve the deployability of PANE. The actions in PANE are inspired by FML [11], which it extends by involving end-users, adding queries and hints, and introducing a time dimension to action requests. Resonance [16] delegates access control to an automated monitoring system, using OpenFlow to enforce policy decisions. Resonance could be adapted to use PANE as the mechanism for taking action on the network, or could be composed with PANE using a library such as Frenetic [7].

Using Application-level Information Many previous works describe specific cases in which information from end-users or applications benefits network configuration, flexibility, or performance; PANE can be a unifying framework for these. For example, UPnP [21] allows applications to control a network gateway, such as to add a port-forwarding entry to a NAT table. ident++ [15] proposes an architecture in which an OpenFlow controller reactively queries the endpoints of a new flow to determine whether it should be admitted. In contrast, PANE allows administrators to delegate the privilege to install restricted network-wide firewall rules, and users can do so either proactively or reactively (cf. §5). D^3 [24] is a transport protocol for datacenter networks that replaces TCP's congestion control with explicit rate control to meet transmission deadlines. End-hosts in D^3 request sending rates based on a flow's size and deadline, and receive explicit rate signals from the routers along the flow's path. We plan to test similar mechanisms in PANE using hints about flowgroup properties.

Networking and Declarative Languages PANE's design is inspired by projects such as the Router Configuration Checker [6] and the Margrave tool for firewall analysis [17] which previously applied declarative languages to network configuration. Both tools use a high-level language to detect configuration mistakes in network policies by checking against predefined constraints. PANE, however, directly integrates such logic into the network controller. NDlog [13] adapted the declarative Datalog language to ease the programming of distributed applications and protocols; PANE receives requests from distributed clients, but their evaluation is centralized.

7 Discussion

Participatory networking makes the programability of SDNs accessible to end-users and their applications. PANE is an initial design, and its deployability depends on mechanisms which provide the desired guarantees. The controls currently provided by OpenFlow are a strong start, and PANE will benefit from new developments, such as the path selection possible in novel datacenter networks, or better QoS mechanisms. PANE's deployment also depends on application adoption, which is largely a question of perceived costs and benefits. A PANE network should not provide worse service than existing networks, and should provide tangible benefits for the right uses. We are encouraged by examples of improved performance and functionality due to application information (*e.g.* [10, 15, 24]). PANE's protocol is designed for simplicity; an undergraduate required less than two days to gain an understanding and add support to Ekiga.

Each of PANE's three components – privilege delegation, interaction protocol, and network controller – is currently under active development. We are investigating requests for resources such as latency, jitter, and path selection, as well as query and hint messages. These pose many exciting challenges, such as request composition, and handling misleading hints. We also intend to evaluate the scalability of the controller. Finally, the participatory nature of PANE is suggestive of a market for network resources, which we intend to develop and which will likely require the design of transactional PANE requests.

Acknowledgments The authors wish to thank Scott Shenker, Jennifer Rexford, and Joe Politz for helpful discussions and feedback, and Andreas Voellmy for augmenting Nettle to support PANE. This work was partly supported by NSF grant 1012060. Andrew Ferguson is supported by a DoD NDSEG fellowship.

References

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proc. NSDI '10*, San Jose, CA, 2010.

[2] H. Bazzaz, M. Tewari, G. Wang, G. Porter, T. S. E. Ng, D. Anderson, M. Kaminsky, M. Kozuch, and A. Vahdat. Switching the optical divide: Fundamental challenges for hybrid electrical/optical datacenter networks. In *Proc. SOCC '11*, Cascais, Portugal, 2011.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475, Dec. 1998.

[4] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994.

[5] D. Erickson, et al. A demonstration of virtual machine mobility in an OpenFlow network. In *SIGCOMM '08 (Demo)*, Seattle, WA.

[6] N. Feamster and H. Balakrishnan. Detecting BGP configuration faults with static analysis. In *Proc. NSDI '05*, Boston, MA, 2005.

[7] N. Foster, M. J. Freedman, R. Harrison, J. Rexford, M. L. Meola, and D. Walker. Frenetic: a high-level language for OpenFlow networks. In *Proc. PRESTO '10*, Philadelphia, PA, 2010.

[8] 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. *SIGCOMM CCR*, 35:41–54, 2005.

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an Operating System for Networks. *SIGCOMM CCR*, 38:105–110, July 2008.

[10] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Trans. Comput. Syst.*, 21:207–233, May 2003.

[11] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker. Practical declarative network management. In *Proc. WREN '09*, Barcelona, Spain, 2009.

[12] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula. Automated and Scalable QoS Control for Network Convergence. In *Proc. INM/WREN '10*, San Jose, CA, 2010.

[13] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking. *Commun. ACM*, 52:87–95, Nov. 2009.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. *SIGCOMM CCR*, 38:69–74, 2008.

[15] J. Naous, R. Stutsman, D. Mazières, N. McKeown, and N. Zeldovich. Enabling delegation with more information. In *Proc. WREN '09*, Barcelona, Spain, 2009.

[16] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance: dynamic access control for enterprise networks. In *Proc. WREN '09*, Barcelona, Spain, 2009.

[17] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi. The Margrave tool for firewall analysis. In *Proc. LISA '10*, San Jose, CA, 2010.

[18] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich. Energy management in mobile devices with the Cinder operating system. In *Proc. EuroSys '11*, Salzburg, Austria, 2011.

[19] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the Production Network Be the Testbed? In *Proc. OSDI '10*, Vancouver, BC, Canada, 2010.

[20] I. Stoica, H. Zhang, and T. S. E. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. In *Proc. SIGCOMM '97*, Cannes, France, 1997.

[21] UPnP Device Architecture version 1.1. UPnP Forum., Oct. 2008.

[22] A. Voellmy and P. Hudak. Nettle: Taking the sting out of programming network routers. In *PADL*, pages 235–249, 2011.

[23] R. Wang, D. Butnariu, and J. Rexford. OpenFlow-based server load balancing gone wild. In *Proc. Hot-ICE '11*, Boston, MA, 2011.

[24] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron. Better never than late: meeting deadlines in datacenter networks. In *Proc. SIGCOMM '11*, Toronto, Canada, 2011.