Getting Students' Hands Dirty with Clean-Slate Networking

Nick Feamster
Georgia Tech

Jennifer Rexford
Princeton University

ABSTRACT

Conventional networking courses treat today's protocols and mechanisms as fixed artifacts, rather than as part of a continually evolving system. To prepare students to think critically about Internet architecture, we created a graduate networking course that combines "clean slate" networking research with hands-on experience in analyzing, building, and extending real networks. Our goal was to prepare students to create and explore new architectural ideas, while teaching them the platforms and tools needed to evaluate their designs in practice. The course, with offerings at both Georgia Tech and Princeton, focused on network management as a concrete way to explore different ways to split functionality across the end hosts, network elements, and management systems. The programming assignments exposed students to a range of systems, including Click, Quagga, Emulab, OpenFlow/NOX, Mininet, the Transit Portal, and publicly-available Netflow and BGP measurement data.

1. Introduction

Most networking courses are organized around the layers of the network protocol stack, implicitly assuming that the Internet architecture is set in stone. Network architect John Day argues that we teach the design of the Internet as a "paragon of engineering, rather than a snapshot of our understanding at the time". Historically, this mode of instruction was a natural reaction to the difficulty in changing the form and function of existing protocols and networking equipment. Recent trends in networking research—notably, the "clean slate" networking philosophy and the increasing availability of programmable network elements—create new opportunities to re-think the Internet architecture. Changes to the Internet could range from minor tweaks to a major refactoring of network functionality.

These trends suggest to us that networking should be taught in a fundamentally different way, but they also raise questions about how to structure a course without the familiar scaffolding of the network protocol stack. Because so many networking concepts are abstract, so we want to present these concepts in a real, practical context and equip students to go beyond high-level architectural ideas to specific designs and implementations. In other words, the course should both train students in clean-slate design *and* get their hands dirty with the details of network protocols and systems. Our new course design aims to achieve these goals through a combination of lectures, selected readings, and programming assignments that focus on specific networking *problems*, rather than any particular protocol, layer, or application.

Organizing around problems rather than layers avoids redundant material while stressing concepts that appear at multiple layers of the protocol stack. For example, the Ethernet back-off mechanism and TCP congestion control are both examples of distributed resource-allocation algorithms running on the end hosts. This organization also allows us to compare and contrast multiple approaches to the same problem. For example, should load balancing over network paths happen through central optimization of the routing-protocol configuration or distributed control where hosts shift traffic to less-congested paths? To scope the material, our first offering of the course focused on problems that arise in *network management*, covering both how to manage today's networks and how to design future networks that are easier to manage. We explore a range of settings, including home, enterprise, data-center, and backbone networks.

A cornerstone of the course is the set of hands-on programming assignments and laboratories that teach students the platforms, tools, and data sets commonly used in networking research. Our goal is to train students to take their architectural ideas beyond paper designs by building and deploying real systems. We also believe that hands-on experience is a better way to learn the considerable domain details in today's networking protocols than marching through the individual protocols in lecture. In the first offering of the course, the assignments covered programmable network elements (*e.g.*, Click [7], Quagga [11], and OpenFlow/NOX [3]), experimental platforms (*e.g.*, Emulab [2], Mininet [8], and the Transit Portal [13]), and measurement data (*e.g.*, NetFlow traffic traces and BGP routing data). Using multiple platforms, each with its own scripting languages and parsing tools, also gave the students the confidence to learn new systems for their own research.

We have taught two instantiations of graduate-level networking courses following the philosophy outlined above, at both Princeton and Georgia Tech, and course reviews have shown that students find this new course design provides them both relevant practical experience and teaches them critical thinking skills that apply both within the context of networking and beyond. As a service to the community, we have made our course materials available on the SIGCOMM education portal Web site.

2. Course Organization

The course focused on clean-slate network architecture, from the vantage point of the challenges facing the operators of enterprise, data center, and backbone networks. The

emphasis on the definition and placement of function is in sharp contrast to the traditional organization of networking courses by protocol layer. We also selected a diverse mix of research papers emphasizing both old and new work, and clean-slate and evolutionary approaches.

2.1 Network Management Tasks

A course on clean-slate network architecture can quickly devolve into high-level discussions of paper designs without any clear way to argue if one architecture is "better" than another. To make the discussion concrete, we need some sort of user in mind to see if some particular architectural decisions benefits or harms the user in some way. Some users, like end users who run applications or developers who write applications, are somewhat removed from the underlying architectural decisions. The protocol designer has a lower-level view, but typically presupposes some overall design of the system by focusing mainly on a single protocol that fits in a larger framework.

Instead, we chose to focus on the network operator who designs and manages a network. The network operator is responsible for composing many protocols and component together to achieve a variety of relatively concrete goals. As such, focusing on the network operators gives us a frame of reference with well-defined problems. In addition, today's Internet architecture was not designed with network management in mind. This gives us a great opportunity to reflect on how early architectural decisions inadvertently induced difficult network-management challenges that new designs could remedy. As such, we organized our course around the primary tasks network operators perform, with a balance of discussion of how to manage today's protocols and mechanisms and how to design future networks that are inherently easier to manage.

Network configuration: Configuring the underlying network elements is the main challenge in network management. In this segment of the course, we explored how enterprise administrators use Virtual Local Area Networks (VLANs [4]) and backbone operators detect configuration mistakes and express local policies using the Border Gateway Protocol (BGP [12]). We also stepped back to explore the recent trend toward logically-centralized control of networks (culminating in the recent OpenFlow initiative), as a possible way to make network configuration simpler and more flexible.

Traffic engineering: Traffic engineering involves adapting routing to the prevailing traffic to balance load and improve performance. This segment of the course explored several approaches to traffic engineering, such as tuning the configurable parameters in conventional routing protocols, distributed load balancing over multiple paths, and oblivious spreading of traffic over multiple intermediate nodes.

Network security: Network administrators devote significant effort to protecting their users and their own infrastructure from attack. In this section of the course, we covered techniques for preventing attacks (*e.g.*, access control and secure routing protocols) and detecting attacks (*e.g.*, intrusion detection systems and anomaly detection). Discussions of security lead to useful discussion about the need for verifiable notions of identity, something missing in today's Internet.

Network troubleshooting: Network operators rely on measurement data to detect and diagnose problems affecting their users. This section of the course gave an overview of network measurement and effective techniques (such as sampling, filtering, and aggregation) for reducing the volume of measurement data. We also covered techniques for inferring network performance and load from limited measurement data, including a rich body of related work on network tomography.

Emerging topics: Inevitably, some topics don't fit naturally into a simple taxonomy, and this is especially true for active areas of research. In this section of the course, we focused on emerging topics, such as energy-efficient networks and network support for online services (such as geo-replicated servers and IPTV). Then, the course ended with a look at programmable and virtualized networks, allowing us to reflect on the ongoing debate in the research community on the merits of rethinking the Internet architecture.

2.2 Rethinking the Division of Labor

The course focused on network *architecture*—the definition and placement of function in the network—rather than the existing protocol stack. How to divide functionality between the end hosts, the network elements, and the management systems was a recurring theme in the lectures, papers, and discussions.

Overview of today's division of labor: To set the stage for the course, we had three lectures—on the end host, the data plane, and the control plane—that reviewed computer networking, with an emphasis on today's Internet architecture. The lecture on the end host focused on how the bootstrapping the end host, network identifiers (*e.g.*, domain names, IP addresses and MAC addresses), the socket abstraction, and distributed resource-allocation techniques like TCP congestion control and the Ethernet back-off mechanism. The lecture on the data plane focused broadly on packet-streaming algorithms that perform simple actions on incoming packets based on previously-installed rules, providing a common framework for thinking about routers, switches, firewalls, network address translators, and so on. The lecture on the control plane discussed routing protocols, with a clear separation between what state the protocol computes (*e.g.*, spanning tree vs. shortest-path graph), the distributed algorithms that compute these paths, and how the routers learn where end hosts connect to the network.

Revisiting the division of labor: The course frequently returned to the theme of rethinking the division of labor. For

example, the section on traffic engineering fostered discussion of whether traffic engineering should be handled by the management system (*e.g.*, tuning OSPF link weights), the network elements (*e.g.*, load balancing in TeXCP [5]), or the end hosts (*e.g.*, by switching paths in response to performance problems). Similarly, the section on configuration in enterprise networks lead to discussion over whether switches should learn end-host locations through MAC learning in the data plane (as in today's Ethernet) or by querying a distributed directory in the control plane (as in SEATTLE [6]).

Multiple types of networks: To give the students broad exposure, we covered a mix of enterprise, data-center, backbone, and home networks; ideally, the course would have also covered wireless networks as well. Considering different kinds of networks fostered good discussions of the unique opportunities and constraints in each network setting. For example, while traffic engineering in data-center networks must handle volatile traffic patterns, greater path diversity and the opportunity to modify the end host lead to different solutions than earlier work in backbone networks. Similarly, in the discussion of security, enterprise network operators place greater emphasis on access control and intrusion detection, whereas backbone operators focus on routing-protocol security and network-wide anomaly detection.

2.3 Diverse Collection of Research Papers

In organizing the course, we intentionally exposed students to a wide range of research papers, including a mix of classic and recent papers, evolutionary and clean-slate research, and interdisciplinary work incorporating relevant theoretical techniques.

Mix of classic and recent papers: As with many graduate courses, we wanted to have a healthy mix of classic papers in computer networking and newer papers reflecting the current activity in the field. The classic papers fit naturally at the beginning of the course (in conjunction with a brief overview of the current Internet architecture) and at the end of the course (as a way to look forward toward a future Internet). Several of the classic papers, written well before the Internet's success was a forgone conclusion, offered refreshing examples of clean-slate design in action, while ironically also providing background on the legacy Internet. Others, such as the early work on active networks, gave the student an opportunity to see how certain themes in network research recur, as witnessed by the renewed interest in programmable networks.

Evolutionary vs. clean-slate solutions: For most classes, we intentionally paired a paper proposing a clean-slate solution with a corresponding evolutionary approach to the same problem For example, the class on access control included one paper describing a formal tool for configuring existing firewalls (Firmato [1]) and another on a dynamic access control system built on top of OpenFlow (Resonance [10]). Similarly, the class on interdomain routing security included incrementally-deployable techniques for detecting and avoiding anomalous BGP routes, as well as clean-slate cryptographic solutions like S-BGP and soBGP. Comparing these kinds of papers fostered to good discussions of the challenges (and successes) of working within today's network architecture, as well as the opportunities for entirely new approaches to network-management problems.

Interdisciplinary research: The course stressed that practical problems in networking often lead to interesting interdisciplinary work that applies theoretical techniques from other fields. For example, the treatment of traffic engineering naturally emphasized connections to optimization theory and control theory, whereas the discussion of network troubleshooting led to discussions about tomographic techniques for inferring network properties from limited measurement data. While we could not cover the many theoretical topics in great depth, the course stressed the value of applying techniques from other fields or collaborating with colleagues in theoretical fields.

3. Programming Assignments

While many graduate courses focus primarily on reading and discussing research papers, we believe that programming assignments are crucial for teaching students tools for use in research and helping them understand the material. In this section, we survey the goals of the assignments and summarize the specific assignments used in our two offerings of the course.

3.1 Goals of the Assignments

Our main goal for the assignments was to *help students develop an arsenal of tools* that they could apply to implement and evaluate their own research ideas. To achieve this goal, the assignments require students to become familiar with a variety of platforms, including development platforms such as Emulab, OpenFlow, and the Transit Portal, but also with tools and techniques for analyzing network measurement data. Each assignment focused on a different platform; our introduction of many different platforms and tools served not only familiarize the students with a variety of tools, but also to improve their confidence in quickly learning new environments for development, testing, and experimentation..

Many networking courses aim to *teach domain details*. Learning the details of various networking protocols is essential, but we believe that the best way for students to learn these details is *by doing*: students learn the details of these protocols best when they can gain exposure to these protocols through hands-on tasks and activities. For example, the abstract details of the Border Gateway Protocol can be made much more concrete if students are asked to establish a live BGP session and direct traffic over that session; similarly, various network operations tasks such as billing and provisioning can be made more concrete if students are given network data and asked to perform the tasks themselves.

Many networking courses teach students networking by presenting the details of each layer of the network stack. In contrast, one of our goals in the assignments is to *teach students to think across traditional layer boundaries*. We believe that many important network concepts and network management tasks transcend layers, and that some network management problems may actually result from interactions across layers. In one of our assignments, we ask students to implement the the same set of network functions at different layers of the protocol stack, effectively demonstrating that the concepts are largely the same with only subtle differences in the details (*e.g.*, the format of forwarding-table entries in a switch vs. a router).

In each of the assignments, we encourage students to think critically about network design—and the design of experimental platforms themselves—with an eye toward redesigning systems to make them easier to use and extend. Admittedly, the assignments could go much further in helping students think critically about design: the current versions of the assignments still focus largely on familiarizing students with the mechanics of existing platforms and protocols. For each assignment below, we describe possible ways to extend it to encourage students to think more critically about network design.

3.2 Summary of Assignments

We now summarize the assignments we developed and describes the conceptual themes, the skills that students develop, and the domain details that students learn when doing the assignment. To expose students to problems in different environments, we have students study problems in both enterprise networks and wide-area networks. We are also in the process of developing an additional assignment for familiarizing students with home networks, which we will discuss in Section 4.

3.2.1 *Enterprise Networks*

Layer-Two Configuration (using Emulab, Click, and Quagga) To familiarize students with layer-two networking concepts, as well as with Emulab and Click, we provided students with a sample network topology and asked them to set up the topology as a single, switched local area network. First, to familiarize students with the capabilities of Emulab, we asked them to use the scripting language embedded in Emulab's configuration to automatically configure a large network topology. We then asked the students to use Emulab capabilities to place all of these nodes in a single local area network. To familiarize students with layer-two networking concepts, as well as the Click software router, we then asked students to implement the same layer-two connectivity, but using a Click configuration, rather than relying on the Emulab infrastructure to provide the layer-two topology. This part of the assignment taught students the details of how learning bridges work and familiarized them with the capabilities and configuration of Click elements. The students also experimented with intradomain routing protocols by running OSPF on the Quagga routing software. This gave the students experience with using and configuring Quagga, and measuring the time required to detect and recover from a link failure.

The current version of this assignment familiarizes students with existing layer two protocols; we intend to extend the current assignment to also encourage students to think more about new and emerging layer-two protocols, such as various emerging scalable Ethernet protocols (*e.g.*, SEATTLE [6], Portland [9]). Although our current course offerings teach many of these designs in lecture, we believe that students may be able to reason about the tradeoffs associated with each of these designs.

Hubs, Switches, and Routers (using Mininet and OpenFlow/NOX) The next assignment illustrates how many kinds of network elements ultimately perform similar rules on streams of packets that involve matching on attributes and performing actions, and exposed students to OpenFlow switches and the Mininet environment. The students downloaded a VirtualBox [14] image that runs Mininet to emulate a network of hosts, switches, and controllers. After constructing a simple topology, the students write a series of small applications that run on the NOX controller. In the first two applications, all packets go to the controller which acts either as a hub or a learning switch. In the next two applications, the controller applications install either microflow rules or wildcard rules in the underlying switches, so the switches act either as a layer-two MAC-learning switch or an IP router. In all four cases, the students perform ping measurements between various pairs of hosts and analyze the sources of delay in delivering traffic.

The assignment intentionally uses the same platform to study hubs, switches, and routers, to stress the inherent similarities between these components and the value of a general data-plane model in network elements. Still, the current assignment primarily focuses on the mechanics of using OpenFlow, NOX, and Mininet. In a future version of the assignment, we could have the students design and implement fundamentally new network functionality on top of the OpenFlow switches.

3.2.2 *Wide-Area Networks*

We have developed two assignments to expose students to problems in wide-area networking. The first assignment asks students to set up a virtual network and connect that network to the Internet via BGP, using the Transit Portal. The second assignment asks students to analyze different network traces, asking them to perform various network management tasks.

Connecting a Virtual Network to the Internet (using Transit Portal) In this assignment, we give students a simple network topology and ask them to set up the topology in a virtual network environment of their choice. We gave students the option of selecting their own virtual network environment to allow students to focus on the networking-related

aspects of the assignment, rather than on the minutia of any particular virtual network environment. Most students opted to use either VirtualBox or Qemu to set up their virtual network. After setting up the network topology, students were asked to run an intradomain routing protocol between the nodes (*e.g.*, by configuring a software router on each of the nodes). All students elected to use Quagga for this part of the assignment. Once students set up an intradomain routing protocol on the nodes, we asked them to connect their network to the Internet via BGP (using the Transit Portal as an upstream provider) and announce an IP prefix corresponding to their virtual network over the BGP session that they established.

Once the students established interdomain connectivity, we asked them to perform or demonstrate a few simple network management tasks. First, we asked them to establish an HTTP server on one of the nodes in the topology and demonstrate that the server was reachable from the Internet. We then asked them to introduce a link failure in the intradomain topology and demonstrate intradomain route convergence. By performing this experiment, students were able to understand interdomain routing configuration and understand how intradomain routing protocol convergence works. Finally, we asked them to rate-limit the HTTP traffic on the link between the virtual network and the rest of the Internet. Most students used the Linux `tc` tool for this part of the assignment.

A future version of this assignment might ask students to perform more network management-related tasks with the network that they set up (*e.g.*, experimenting with different traffic load balancing schemes).

Analyzing Measurement Data (using Netflow and BGP Traces) We gave students both routing and traffic traces from several real-world networks and asked them to perform several management-related traffic analysis tasks using the data. The assignment included a BGP routing table dump, as well as NetFlow traces from the Georgia Tech and Internet2 networks. In the traffic measurement portion of the assignment, we asked students to analyze network flow traces. We asked students to address various network management-related questions, such as determining the heavy-hitter applications and users for both inbound and outbound traffic. We also asked them to propose methods for balancing traffic load across the available links. In the BGP portion of the assignment, we asked students various questions about the BGP routing table, ranging from which ISPs served as the upstream provider for various edge networks to how various networks perform inbound and outbound route control. We also asked the students to analyze streams of BGP update messages to under BGP convergence behavior, such as the frequency of updates across different prefixes, the interarrival times between messages during BGP path exploration, and typical convergence delays.

In future offerings of the course, we aim to better integrate this assignment with the experimental tools and platforms that students use for other assignments. One possibility might be to integrate the BGP-related questions with the assignment that uses Transit Portal, and perhaps even have students set up traffic monitoring tools on the virtual network that they establish.

4. Discussion and Lessons Learned

In this section, we briefly discuss the lessons we have learned from our course offerings, and what we plan to do differently when the course is offered in the future. Many of the discussion points below reflect the feedback that we received on course feedback forms collected at the end of the semester at both Princeton and Georgia Tech. Most students expressed excitement at the new course organization; one even told us that the course modules on OpenFlow helped him better prepare for a job interview for a job that he ultimately landed.

Alternate Course Organization In the first offerings of the course, we organized lectures around network management tasks, in an attempt to eliminate the redundancy in instruction that can occur when networking is taught layer-by-layer. Still, the existing organization also had problems, since different network domains (*e.g.*, data-center networks) were spread throughout the course, rather than consolidated in a single module. This organization made it more difficult to align the assignments thematically with lecture topics, since each assignment naturally focused on one particular domain. It also introduced some abruptness into the flow of lectures, since some lecture topics naturally required introducing more background material before an in-depth discussion of the research problems could take place. In the next offering of the course, we will attempt to organize the lectures around different types of networks (*i.e.*, enterprise networks, data-center networks, transit networks, home networks, and wireless networks) in an attempt to streamline the lectures and align them more closely with assignments.

More Novelty, Less Grunt Work In course surveys, many of the students complained that many of the assignments involved a substantial amount of "grunt work" to set up the infrastructure required to do the assignments. Some of this grunt work entailed setting up the platforms and infrastructure associated with the assignments and is, in some sense, the nature of becoming familiar with the ins and outs of new tools and platforms for networking research.

Nevertheless, we believe that some aspects of assignment setup could be better streamlined, to help students focus on the more interesting aspects of each assignment. For example, in the assignment that requires students to build a learning switch with Click on the Emulab testbed, students spent a considerable amount of time finding and compiling a version of Click that would run on the default Emulab image. One way to streamline this process might be to create an Emulab image that students can install on the Emulab nodes that already has some of the working software for the course

(*e.g.*, Click, Quagga). In the absence of teaching assistant support, another possibility will be to openly encourage students to work together on the aspects of the assignment that involve experiment setup. For example, one idea we may explore in the future is to use a course wiki to allow students to collaborate and troubleshoot certain parts of the assignment (*e.g.*, the initial platform setup) together as a class.

Another possibility that we are exploring is to design assignments that build on one another. This approach may not be possible as a general rule, because, by their nature, the assignments are intended to teach students about a variety of platforms for networking research, but there may be some cases where tools or concepts learned in one assignment might be re-used in another. For example, after students learn about Click elements and configuration in the initial assignment on layer-two networking, we might ask them to re-use Click in the assignment on building a virtual network to perform specific network management tasks (*e.g.*, to implement the rate-limiting function, or some other traffic classification function, in Click, rather than with `tc`).

New Course Modules In keeping with a new organization of course material around different types of networks, we plan to augment the course with material on an emerging fourth type of network: home networks. In the additional module and planned assignment, we intend to explore the theme of network-management problems that occur in this new increasingly important type of network. To help students develop hands-on experience with networking problems in the home, we have developed a home networking laboratory where students can install customized (and OpenFlow-enabled) programmable gateways and also create network configurations that include a variety of consumer devices, ranging from game consoles to networked television sets. We are still in the process of developing the assignment for this part of the course, but possibilities including developing an automated network management tool for certain home network configuration scenarios, or possibly a remote troubleshooting module.

On a related note, many of the students asked for coverage of wireless networks. Given the growing prevalence of wireless and cellular networks, future offerings of the course will incorporate a module on these topics, with a course assignment devoted to these topics, as well.

5. Conclusion

The emergence of many new networking technologies in recent years has enabled researchers, developers, and network operators to solve conventional networking problems in radically different ways. Rather than teaching students about network protocols and layers as fixed artifacts, we believe that networking courses should instead focus on the most pressing problems that network users and operators face today, as well as the tools that we, as researchers, can apply to tackle these problems. Towards this goal, we have designed a networking course to help students take advantage of these new opportunities by allowing them to solve problems in networking by completely re-thinking network design and providing them with the tools that they need to implement their designs.

We have found that such a course can better help students appreciate the problems and solutions in today's communications networks, while still teaching students fundamental skills and concepts that will persist in communications networks well beyond today's problems. Indeed, although layering is a fundamental design principle for today's networks, we believe that it need not and should not form the basis of course organization. In fact, many emerging network designs make layers of the protocol stack less distinct, and interaction across layers is also becoming more commonplace. Ultimately, we may find that the network management problems around which we center our course are much more persistent than even the layers of the protocol stack.

Acknowledgments

We thank Yogesh Mundada, Vytautas Valancius, and Richard Wang for their help developing several course modules and assignments.

REFERENCES

[1] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In *Proc. IEEE Symposium on Security and Privacy*, pages 17–31, Oakland, CA, 1999.

[2] Emulab. `http://www.emulab.net/`.

[3] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, July 2008.

[4] IEEE 802.1Q - Virtual LANs. `http://www.ieee802.org/1/pages/802.1Q.html`, 2006.

[5] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005.

[6] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A scalable ethernet architecture for large enterprises. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug. 2000.

[8] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks (at scale!). In *Proc. HotNets*, October 2010.

[9] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A scalable fault-tolerant layer2 data center network fabric. In *Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[10] A. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance: Dynamic access control in enterprise networks. In *Proc. Workshop: Research on Enterprise Networking*, Barcelona, Spain, Aug. 2009.

[11] Quagga software routing suite. `http://www.quagga.net/`.

[12] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, Jan. 2006. RFC 4271.

[13] V. Valancius, N. Feamster, J. Rexford, and A. Nakao. Wide-Area Route Control for Distributed Services. In *Proc. USENIX Annual Technical Conference*, Boston, MA, June 2010.

[14] VirtualBox. `http://www.virtualbox.org`.