Scaling the practical education experience

Joel Sommers
Colgate University
jsommers@colgate.edu *

Andrew W. Moore
University of Cambridge
andrew.moore@cl.cam.ac.uk

ABSTRACT

In the past decade, a number of environments to support hands-on exercises for undergraduate- and graduate-level networking courses have been developed. In this position paper, we consider the successful model of a graduate-level laboratory-based course, and how similarly compelling hands-on experiences may be brought to a wider audience at the undergraduate level. While there are significant challenges to adapting the advanced content to an undergraduate audience, there also exist great opportunities for compelling hands-on activities. We discuss the promising aspects and potential perils of this approach, suggesting some possible ways forward.

1. INTRODUCTION

Teaching and learning are best approached using a spectrum of methods. There can be little argument that a range of pedagogical practices generally leads to the best educational outcomes. Considerable discussion has taken place in the networking community regarding how best to engage students in practical, hands-on experiences [10], and indeed many compelling approaches have been developed over the years. In this position paper, we revisit the perennial question: how do we get students *trained* as well as *educated*?

We acknowledge the considerable work in the Computer Networking community invested in delivering practical material to students, and will discuss several examples of it through the course of this paper. Yet, we recognise issues that remain as roadblocks between where we currently are, and where we might wish to be. Moreover, we recognise the need for greater sharing and dissemination of successful approaches, and adaptation of those approaches to wider audiences.

Integration of practical, hands-on, and experimental work into an undergraduate or graduate-level networking course is challenging. We do not claim to offer a panacea, or even a particularly novel approach. Rather, we are like "dwarfs on the shoulders of giants", or more coarsely put, educational opportunists who seek to build-on and adapt the work of the community at large. In this paper, we outline a successful graduate module not as the ideal and definitive goal but as an example of a graduate programme that any institution might wish to host. Alongside an outline of this module, we discuss the number of forms this module has taken as it has been adapted between various institution and education-settings. However, we treat this module as a destination example and then focus on how the *head-pressure* of background material and experience necessary for a typical candidate student may be provided within the Computer Network context.

We argue that students at both the undergraduate and graduate levels should engage in practical exercises that enable them to grapple with important and realistic scientific/engineering issues in networking. Moreover, we view the "Building an internet router" (BIR) course described in § 3 as one successful example of a set of coherent practical exercises, though geared presently toward advanced/graduate students. We advocate broadening this approach to include other best practices (*i.e.*, develop and extend an body of practicals). In this paper we focus on the challenges and opportunities of scaling/adapting the BIR approach to a broader audience. We first discuss existing approaches to hands-on activities in undergraduate-level courses. We then describe the BIR course in some detail. Lastly, we discuss challenges and opportunities in adapting and broadening the approach of BIR to include undergraduate institutions.

2. CURRENT UNDERGRADUATE APPROACHES

In this section we briefly describe and characterize various approaches that have been taken in undergraduate networking courses to provide "hands-on"-type experiences for students. We divide the approaches into two categories: those primarily employing simulation and/or trace-based exercises, and those involving experiment in a laboratory setting, using either entirely "real" components, or some emulated components. Note that we do not claim to be comprehensive in this section; the references are merely illustrative of approaches that have been taken and described in the literature.

We will not enter a description of other (relevant) aspects of computer science education such as the teaching of programming languages or execution or assessment of group projects. Such aspects have been well studied elsewhere, for

*This work was done in part while Joel Sommers was visiting the University of Cambridge, Computer Laboratory.

example Fincher *et al.* provide an excellent study of the pedagogy of computer science group projects in [8].

2.1 Simulation and trace-based approaches

Simulation and trace-based exercises generally involve students using software such as ns2 [16] to construct simulated network topologies and to conduct traffic, routing, or other kinds of experiments. Indeed, a repository of ns2 scripts has existed for many years for educators to post and share simulation scripts for examining different networking scenarios [1].

Simulations remain a popular choice for creating highly constrained experiments for students to examine particular network behaviors by modifying parameter settings. For example, the approach described in [9] uses the Opnet simulator in experiments to examine the effect of turning Nagle's algorithm on or off, or the TCP receiver's window size.

Simulation experiments can be valuable for demonstrating and investigating particular network behaviors. The ability to quickly rerun a simulation with different parameter settings make this technique appealing. Moreover, some parameters can be difficult to modify and/or control in a more realistic environment. However, using a simulator such as ns2 in a more general way is problematic, since it is a highly complex piece of software with a steep learning curve. A number of efforts have attempted to create simplified simulation settings specifically for educational purposes, *e.g.*, the Netwire system [3]. This system enables students to more easily write network simulation programs that go beyond simple parameter modification.

Because no "real" network traffic is generated and no live networking components are invoked during experiments, we classify activities that involve analysis of previously captured packet traces with a tool like Wireshark along with simulation-based activities. Exercises involving packet trace analysis can provide students a flavor of realism without the overhead of developing and maintaining laboratory infrastructure, and without the risks involved in having students generate traffic in a live setting [9, 15]. Trace analysis exercises are often found in networking textbooks, since it can be generally assured that all readers can easily download and run the necessary software to perform the trace analysis.

2.2 Laboratory and emulation-based approaches

In the last several years, a number of hands-on experiences based on dedicated laboratory equipment that use either real or emulated network components have been explored.

At one end of the spectrum, [6] describes a set of hands-on experiences for students using commodity PCs running Linux, and other free and open source software such as Wireshark, Snort, and the Zebra router. [19] also describes the use of an environment that relies on the Zebra router and a set of scripts to configure hosts in particular logical topologies for use in experiments. Similar to these approaches is the one described by Pan [18] that centers on using the Linksys WRT54GL and the OpenWRT firmware to provide students with a realistic platform for experiments.

Hands-on settings that make use of the Emulab[2] software have seen fairly wide use in networking education. These settings allow students to create virtual topologies using a set of commodity hosts, and to emulate different link characteristics such as delay and loss. These settings enable a fair amount of realism, while allowing lab resources to be multiplexed among students. Laverell *et al.* [11] describe their experiences setting up and using the Emulab software in networking courses. The Tinkernet system described in [12] contains functionality similar to that in Emulab. The authors of that paper describe a set of laboratory experiences that focus on building a networking protocol stack.

While Emulab-type systems virtualize links, other systems present entire virtual network topologies to students, enabling different levels of interaction with the virtual network. For example, the GINI system [14] creates virtual networks using tunnels and Linux virtual machines, enabling students to use standard tools and APIs to interact with the virtual network. Similarly, the Virtual Distributed Ethernet system [13] emulates an Ethernet switch and allows virtual and real machines to be stitched together into topologies that students can interact with using standard tools. The Virtual Network System (VNS) [4] also allows students to construct virtual network topologies, and allows students to safely interact with their virtual network as well as the rest of the Internet to send and receive raw packets. As a result, students can design and develop a simplified Internet router that runs in user space. The Clack Graphical Router [5] can be used in conjunction with VNS to construct topologies and visualize different network behaviors. Finally, the Open Networking Laboratory [20, 21] enables students to remotely access and control real networking hardware in a laboratory setting. It provides many knobs for controlling router and system behavior, and for visualizing and experimenting with different network scenarios. Students can create plugins to modify packet processing behavior in a router.

2.3 Discussion

The form and scope of practical experiences for undergraduate networking courses has evolved since the most recent SIGCOMM education workshop. In particular, it was noted in the workshop report [10] that many laboratory-based courses had only recently been introduced. Since that time, quite a number of laboratory-style hands-on approaches have been advocated and developed, some of which are described above. Indeed, the trend in recent years seems to be toward developing more realistic hands-on exercises for students.

While the trend toward more compelling practical exercises is a development that has likely led to higher student engagement and satisfaction, these improvements have not

[1] `http://www.isi.edu/nsnam/repository/`.

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

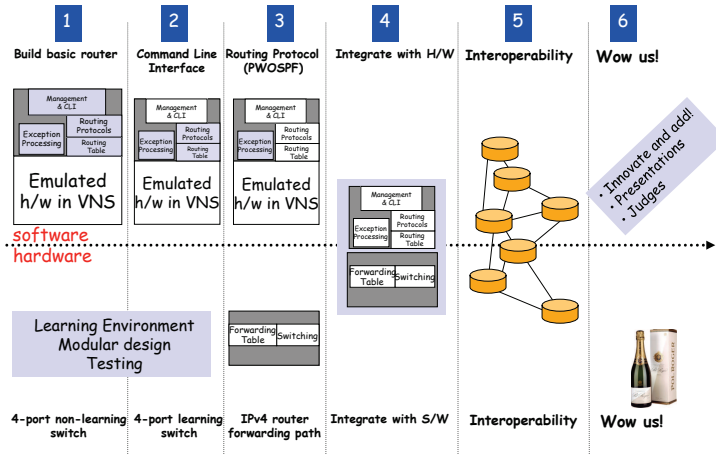

Figure 1: *Building an Internet Router*: **an outline (based upon Nick McKeown's CS344 introduction slides)**

come without tensions or difficulties. In particular, more realistic laboratory-type exercises can be difficult to scale to a large number of users, both in terms of the underlying emulation or laboratory management infrastructure [11], and in terms of assessment of student projects. The development of purpose-built virtualized environments can be seen as a perhaps inevitable response to the problems of scalability, yet operating any medium–large scale experimental environment is likely to continue to be challenging.

With respect to assessment, the scalability problem tends to tip the balance in favor of more narrowly constrained exercises in order to make grading easier. Unfortunately, if exercises are "too" constrained, they can devolve into activities in which students simply go through the motions. Another problem with scalability may be that there are simply not enough tutoring resources to support even modest class sizes at smaller undergraduate institutions. This problem can be particularly acute when students enter the networking course with a weak systems or programming background, which can frequently happen at smaller institutions. For these reasons (among others), simulation and trace-based exercises remain compelling for certain types of investigations. On the other hand, relying too much on more easily constrained (and graded) simulation or trace-based exercises is often seen by students as less exciting, and less directly tied to developing and improving marketable job skills.

3. BUILDING AN INTERNET ROUTER

The *Building an Internet Router* (BIR) module: the CS344[3] course at Stanford University, and it's progeny including the P33 module at Cambridge University[4] is a graduate-level subject. Offered as part of Master's programme but also available to advanced undergraduate and Ph.D. students, the BIR module is delivered as a project-based subject executed by a small team of students (commonly two). There is considerable flexibility in the operation of the module as well as additional differences between each Universities' offering (some of these are detailed in § 3.1).

Educational intent

From the outset BIR sets out to achieve a number of education goals: as a piece of group work it is not practical for one group-member to perform macho-programming and write all the code for all group members. In this way BIR is a great example of what might be achieved when a project is sufficiently modular. Alongside any large software project, BIR provides a natural environment for wider-ranging skills such as version repository and continuous design documentation to be executed. Alongside these skills, BIR has a number of computer networking specific outcomes. Hardware work covers issues of good algorithm and data-structure design (particularly if the team seeks a router capable of the highest throughput), as well as a principled approach to designing a significant amount of hardware (Verilog). Both the hardware and software sides of the project need to employ unit testing and the provision of a suitable test framework enables such consistent hardware testing throughout development. The software side to this work faces a range of problems common in any real-time system, issues of concurrency, the optimum implementation of timers and multiple threads of execution are the starting point. Understanding the issues of implementation both routing protocols and more common-place mechanisms such as ARP become critical to Alongside these, the software team-member must consider his approach to verification/test-suites as well as how testing with other groups will take place. All of this is alongside the practical application of material learnt in subjects with titles such as "Digital Communication" or "Computer Networks" where details of protocol abstraction and packet format may first be presented.

BIR has another aspect that is critical: it is extremely rewarding. For example, an early stage in development achieves functional connectivity and students find the experience of connecting a regular browser to the web-server-instance (hosted within VNS) — using their software router to pass packets — particularly gratifying. Working IPv4 routers can be connected to the Internet and students often exclaim how much pleasure they are given seeing their work (hardware and software) working as it should. In this way, the philosophy behind BIR also embraces a crtical aspect of Computer Networking. From the outset it is emphasised that for routers to be useful in the Internet they must interact with other routers; this leads to the groups also needing to communicate.

BIR in detail

BIR is run as a group project; the high-level outcome is that each group build a working Internet IPv4 router *and* design and implement advanced functionality that extends the basic IPv4 system. Because the project is done as a combination of hardware and software, each team can create a fully

[3] `http://yuba.stanford.edu/cs344/`
[4] `http://www.cl.cam.ac.uk/teaching/1011/P33/`

IPv4 router capable of 8Gb/s. The module is designed for student teams of two: one student takes on the hardware aspects while the second student dedicates themselves to the software.

The design of CS344 enables the project to proceed (for some weeks) independently of setbacks for either of the team members. In no small part this occurs because of a huge investment in infrastructure isolateing development of hardware and software work and permitting simultaneous progress: this is a sophisticated test and verification framework that encourages focused hardware development and, through the VNS infrastructure[4], completely independent software development and testing. VNS permits students to develop and run software from any suitable Internet-connected computer in a simulacrum of the eventual router environment. This permits all members of each team to operate unencumbered by each other.

The students start the project with sufficient hardware and software to make a system that is not-quite-functional. The hardware itself is based about the NetFPGA card[5], programmable in an HDL such as Verilog while being capable of full line-rate performance. Starting with the Verilog HDL code for a simple two port switch the hardware designer will extend/modify/discard this code to provide the functionality of a four-port IP router. A set of tools and test harnesses are provided to assist the student with design, verification and synthesis.

Alongside the hardware development, the software team-member works with VNS, a click-based[7] environment, that permits students to create, develop and debug the code needed to control and manage the router. The software system as designed and implemented must participate in a dynamic routing protocol and respond to ARP and ICMP messages — a fundamental part of the specification for any Internet router.

The two team-members must then combine the software and hardware systems to create a single complete artifact. The progression is shown in Figure 1 and the integration of hardware and software progresses smoothly by encouraging all groups to use a number of common standards such as the access procedures and layout for the registers by which hardware and software communicate. Use of a common set of standards also permits teams to test against each other and against prebuilt software and hardware *binary* solutions.

The software team-member implements a complete routing protocol. PW-OSPF, based upon OSPF is cut-down to reduce complexity. The students are provided a mock RFC-like document and expected to implement a complete and operational system. Like many RFCs, crucial decisions are left to the implementer or simply omitted. Significant encouragement is given to demonstrating inter-operability among team-members and thus during the course of implementation teams are encouraged to talk through potential incompatibilities in their protocol implementations. It is reasonable (although uncommon) that all the teams of a particular module may converge to their own interpretation of the PW-OSPF specification — one that may not be at all compatible with the pre-canned binaries.

Once teams have unified hardware and software subsystems an official *Interoperability* event is held to permit testing of and between each teams implementation. Once interoperability is demonstrated, each team is considered to have delivered a working IPv4 router; at this point the teams focus upon their advanced feature.

An advanced feature will extend the basic functionality of the IPv4 router. There is a considerable range in complexity and this permits groups to match their skills (and available time) to the size of the task. While a base assessment is made of the advanced functionality, the majority of marks are awarded on the basis of a demonstration and presentation of the advanced functionality. Past features have included a proper firewall, implementations of configuration web-server for configuration, load-balancer, multi-path routing with fast-recovery, and the implementation of VPN endpoint appliances.

Coping strategies

Administration of any group-project-based subject is fraught with issues of differing student ability as well as the need to provide a fair allocation of reward for investment of effort. Considerable effort is invested to ensure balance between software and hardware components. Alongside these the use of extensive weekly check pointing to ensure smooth progress irrespective of the varying skill-levels among students. With teaching assistants and the module facilitators' observations of groups for individual participation and contribution along with a jointly written document submitted to describe the relative contributions of each group-member, addresses issues of fair recognition of contribution among group-members.

One further way we assist students who may not quite match the ideal background for this subject is to provide them access to a computer-assisted ECAD/Verilog learning tutorial. While not a universal panacea the on-line tutorial provides an insight into ways this module might be made more-widely available, particularly to those students that don't posses an optimum background.

3.1 Viva le différence

The Stanford BIR and the Cambridge version have developed in different evironments; we describe how the Stanford module was adapted for Cambridge.

From the outset BIR seemed an ideal module to adapt and run at Cambridge; the coursework Masters programme at Cambridge is targeted students at the same level and students entering this programme might be considered comparable. However, one problem arose quickly: the term-period in Cambridge is shorter than in Stanford: eight taught weeks with six official days of lectures and an average contact loading about 50% higher at Cambridge. For example it is com-

[5] `http://www.netfpga.org`

mon for a Stanford student to limit themselves to only one project module, such as CS344, per-term. While in Cambridge the degree takes place over a single academic year so most students will take multiple project-based modules to accrue sufficient credits.

It is tempting to slice parts of the project from the end, removing the work associated with building advanced router extensions, but this has been considered the most valuable part of the offering at Stanford. Additionally, from an assessment perspective, student-teams differentiate themselves and their work most clearly in these team-specific contributions.

A second solution could be to create bigger teams in the hope that more work will be done in parallel and thus the course material is still possible in it's entirety. Unfortunately it is clear a project might fall-victum to Brooks-law [2] and while groups of different sizes have been tried, this idea was quickly abandoned.

A further approach became the key solution: *do less* by *starting with more*. For example, inspecting the milestones it is clear that a number of modules such as the CLI are easy to isolate, may be considered superfluous to the core objects of the module so these can be provided from the outset.

A more challenging problem occurs with the base hardware language. At Stanford, as is common through the US, many undergraduate students may be taught Verilog, this is also true in Cambridge. However, elsewhere in the world including most of the Universities in Europe, VHDL is more common. Despite these two hardware description languages being considered nearly equivalent (with a number of translation scripts are available), intellectually the equivalent of how learning Pascal instead of Haskell or C instead of Fortran can impose a road-block.

Smooth running of this module has relied on expert teaching-assistant support, a bit of judicious recruiting becomes critical, and past *graduates* of the module provide an important (automatic) renewal process.

Translating the Stanford offering to Cambridge highlighted several significant similarities and a number of differences in the postgraduate offerings. We hope others that may consider adopting this module and teaching methodology will find these insights helpful.

The Cambridge interpretation of Stanford's BIR is neither unique nor the first; COMP519[6] at Rice University had the overt intention of providing practical experience to students of a Network Systems Architecture subject to compliment the taught (lectured) material. The offering is made to students at an earlier stage in their education and some success has been achieved by significantly curtailing the quantity of material in the course, notably the removal of the Advanced feature, and the use of pair-programming for both hardware and software: leading to teams of four students.

3.2 Future of BIR

[6] `http://comp519.cs.rice.edu`

At Cambridge and Stanford the future of BIR is to migrate to the next generation NetFPGA hardware. The challenges associated with building a high-speed router adds to the excitement and interest in this subject. As well as providing closer integration with the standard FPGA build tools, the new NetFPGA hardware improves all the specifications of the previous platform: providing $4\times$10Gb/s ports as well as a larger FPGA fabric. A larger fabric will permit a wider-range of projects and, in time, an IPv6 reference router design.

However, this is only part of the BIR outcome; students gain from their exposure to a complete array of network and systems issues alongside the challenge of group working and ensuring compatible systems.

4. OPPORTUNITIES

Considering the current state of practical experiences for undergraduates, and the advanced, yet highly compelling and successful BIR, we now discuss several challenges and opportunities. We do not uncritically accept the BIR model as the end-all-and-be-all of laboratory-style education experiences. Yet, we do view it as a successful example of a coherent set of practical exercises pitched at the highest levels of thinking skills and educational objectives.

Indeed one of the key challenges with adapting the BIR model to undergraduates is the gap in educational objectives between typical undergraduate- and graduate-level networking courses. *Building an Internet Router* is primarily geared toward higher levels of Bloom's taxonomy [1], assumes that students enter with foundational comprehension of networking concepts, and further assumes some level of prior applied skill development in networking. Also assumed is some level of exposure to hardware design with Verilog (or VHDL). Institutions that are primarily geared toward undergraduates, including liberal arts institutions generally do not have the resources to support a hardware design course. Furthermore, considering that the majority of primarily undergraduate institutions can only offer a single upper-level course in networking, most undergraduates would simply be unable to handle the workload (cognitive and otherwise) required in the BIR course in a single semester.

Nevertheless, we see windows of opportunity. First, one can consider the range of activities employed at the undergraduate level within a taxonomy of (1) *passive learning* (lecture), (2) *observation or demonstration*, (3) *constrained experience* (hands-on demonstration, with the ability to modify parameters, for example), and (4) *unconstrained experience* (hands-on project with a large degree of autonomy). These categories align well with Bloom's taxonomy, and we believe that there is great potential to develop and adapt a coherent set of activities modeled on the BIR and NetFPGA systems that enables students to engage in serious engineering and scientific issues.

For example, the BIR course as conceived can be seen as an example of an *unconstrained* (or loosely constrained)

practical experience. Yet even for undergraduates, one could imagine incorporating more open-ended exercises such as development of an innovative addition to the basic router, *e.g.*, firewall functionality, QoS-aware flow processing, or novel built-in support for traffic measurement.

The majority of undergraduate-style projects are pitched at the *constrained experience* level. For example, one can consider the simulation and visualization of a congested queue, with the ability to modify parameters such as the buffer size as one example of a constrained experience. This example is easily supported on the NetFPGA platform, and other components of the BIR course (*e.g.*, ARP processing, or IP longest prefix match lookup) could be suitably modified to be appropriately limited for smaller-scale undergraduate projects. Indeed, we intend to develop and make available a set of modules for undergraduates modeled on BIR in the near future. We also note that depending whether an instructor chose to use a software-only approach or a combined hardware-software approach, some aspects of BIR could omitted. On the other hand, there are opportunities to incorporate educational technologies and tutors developed by others. For example, the Verilog tutor[7] discussed in [17] could be employed to introduce and include basic hardware design modules into an adaptation of BIR. Clearly, a key to making such an approach successful at the undergraduate level will be to provide appropriate scaffolding and support for students who may have weaker programming or systems skills.

Practical problems related to adapting and deploying the BIR model to undergraduates involve purchasing, setting up, and maintaining some level of laboratory infrastructure for students to use. We note here simply that open access laboratories can help address the lack of access to equipment, that management software/infrastructure such as Emulab can be usefully employed to mitigate these challenges, and that such laboratories and software have improved significantly over the past decade. Still, as noted in [11], laboratory environments continue to be costly and challenging to run and support, and increase instructor overhead.

Lastly, a formidable challenge is that of developing effective assessments. While assessment and the dark art of grading pose challenges in many contexts, they are particularly difficult in loosely constrained settings and in settings with significant technical complexity. While we do not yet have concrete proposals to offer, we intend to offer a set of loosely constrained projects for undergraduates modeled on BIR in the near future, along with a set of clear rubrics to aid in grading.

5. CONCLUSION

In common with most any technology-based discipline, education in computer networking is marked by the need to evolve continuously. In the past decade we have witnessed significant changes in the scope and nature of hands-on experiences for networking students. This continuous change in turn forces academics to maintain constantly changing material and we acknowledge our suggestions only add to the burden. It is through shared material and the courage of colleagues who are willing lower the bar to entry for us by their sharing, that we all can strengthen the enterprise of teaching and learning.

[7] `https://www-ecad.cl.cam.ac.uk/`

5.1 Thanks

Thanks to Nick McKeown, Adam Covington, David Underhill, Glen Gibb, Jeff Shafer, David Erickson, John Lockwood, David Miller, Malcolm Scott, Gordon Brebner, Patrick Lysaght, and the generous support of Xilinx.

This work is supported in part by NSF CAREER award CNS-1054985 and Colgate University.

6. REFERENCES

[1] B. Bloom. *The Taxonomy of Educational Objectives: The Classification of the Educational Goals*. Longman Group Ltd., 1956.
[2] F. P. Brooks, Jr. *The mythical man-month (anniversary ed.)*. Addison-Wesley Longman Publishing Co., Inc., 1995.
[3] E. Carniani and R. Davoli. The netwire emulator: a tool for teaching and understanding networks. *SIGCSE Bull.*, 33:153–156, June 2001.
[4] M. Casado and N. McKeown. The virtual network system. *SIGCSE Bull.*, 37:76–80, February 2005.
[5] D. Wendlandt *et al.*. The clack graphical router: visualizing network software. In *Proceedings of the 2006 ACM SoftVis '06*, 2006.
[6] D. Yuan *et al.* An instructional design of open source networking laboratory and curriculum. In *Proceedings of the 10th ACM SIGITE*, 2009.
[7] E. Kohler, *et al.* The click modular router. *ACM Trans. Comput. Syst.*, 18:263–297, August 2000.
[8] S. Fincher, M. Petre, and M. Clark. *Computer Science Project Work: Principles and Pragmatics*. Springer-Verlag, January 2001.
[9] V. Y. Hnatyshin and A. F. Lobo. Undergraduate data communications and networking projects using opnet and wireshark software. *SIGCSE Bull.*, 40:241–245, March 2008.
[10] J. Kurose *et al.* Workshop Report: ACM SIGCOMM Workshop on Computer Networking: Curriculum Designs and Educational Challenges. *ACM SIGCOMM CCR*, 32(5), November 2002.
[11] W. D. Laverell, Z. Fei, and J. N. Griffioen. Isn't it time you had an emulab? *SIGCSE Bull.*, 40:246–250, March 2008.
[12] M. Erlinger *et al.* Tinkernet: a low-cost networking laboratory. In *Proceedings of the sixth ACE*, 2004.
[13] M. Goldweber, *et al.* Vde: an emulation environment for supporting computer networking courses. In *Proceedings of the 13th ITiCSE*, 2008.
[14] M. Maheswaran, *et al.* Gini: a user-level toolkit for creating micro internets for teaching & learning computer networking. In *Proceedings of the 40th ACM SIGCSE*, 2009.
[15] J. N. Matthews. Hands-on approach to teaching computer networking using packet traces. In *Proceedings of the 6th ACM SIGITE*, 2005.
[16] S. McCanne, S. Floyd, K. Fall, K. Varadhan, et al. Network Simulator ns-2, 1997.
[17] S. W. Moore and K. Taylor. An intelligent interactive online tutor for computer languages. In *25th BCS SGAI*, Nov. 2005.
[18] J. Pan. Teaching computer networks in a real network: the technical perspectives. In *Proceedings of the 41st ACM SIGCSE*, 2010.
[19] S. Yoo *et al.* Remote access internetworking laboratory. In *Proceedings of the 35th ACM SIGCSE*, 2004.
[20] C. Wiseman, K. Wong, T. Wolf, and S. Gorinsky. Operational experience with a virtual networking laboratory. *SIGCSE Bull.*, 40:427–431, March 2008.
[21] K. Wong, T. Wolf, S. Gorinsky, and J. Turner. Teaching experiences with a virtual network laboratory. *SIGCSE Bull.*, 39:481–485, March 2007.