

Teaching Experiences With a Virtual Network Laboratory*

Ken Wong, Tilman Wolf[†], Sergey Gorinsky and Jonathan Turner

Applied Research Laboratory
Washington University in St. Louis
St. Louis, MO 63130
{kenw,gorinsky,jst}@arl.wustl.edu

Department of Electrical and Computer Engineering[†]
University of Massachusetts
Amherst, MA 01003
wolf@ecs.umass.edu

ABSTRACT

The *Open Network Laboratory* (ONL) is an Internet-accessible virtual laboratory facility that can deliver a high quality laboratory experience in advanced networking [3,7]. Our experience with ONL indicates that it has potential to improve student understanding of fundamental networking concepts and increase enthusiasm for experimentation with complex technology. Furthermore, these benefits can be delivered with less effort from the instructor than using a traditional approach of socket programming and ns-2 simulation exercises. The system is built around a set of high-performance, extendible routers which connect personal computers acting as end systems. Users configure their virtual network through the *Remote Laboratory Interface* (RLI), an intuitive graphical interface. The RLI's real-time charts and user data facility make it easy to directly view the effect of system parameters on traffic behavior. These features can enhance learning by providing users with multiple representations of network phenomena. We describe how the ONL facilities have improved our ability to meet instructional objectives and discuss some approaches to improving the laboratory experience.

Categories and Subject Descriptors

K.3.2 [Computers and Education]: Computer and Information Science Education, C.2.6 [Networking]: Routers

General Terms

Experimentation, Human Factors, Measurement

Keywords

Experimental Computer Science, Education, Laboratory

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SIGCSE'07, March 7–11, 2007, Covington, Kentucky, USA. Copyright 2007 ACM 1-59593-361-1/07/0003...\$5.00.

1. INTRODUCTION

High quality advanced technical education is essential for attracting and retaining qualified students, and producing well-educated graduates who can contribute to the development of new generations of technology. Laboratory experiences play a central role in an advanced technical education. They allow students to see how concepts can be put into practice, enabling them to appreciate the real-world implications of what can seem at first very abstract. They also have the potential for increasing the synergy between research and education.

Although the importance of laboratory work is taken as an article of faith by university faculty, the quality of laboratory experiences is highly variable. Several factors contribute to this variability:

- Designing effective exercises is very demanding.
- Maintaining high-quality facilities is expensive and time-consuming.
- Laboratory exercises are prepared with little attention paid to learning research or systematic sharing of experiences.

The Open Network Laboratory (ONL) is an Internet-accessible, virtual laboratory facility for advanced networking that can help to address the issues listed above. First, it can reduce the high expense of a dedicated facility through resource sharing, making it unnecessary for multiple institutions to each have their own staff and facilities. Second, its intuitive user interface can reduce the obstacles that students must overcome to meet educational objectives. Third, faculty at different institutions can share their experiences with their colleagues through the virtual laboratory's common experimental environment. Laboratory facilities such as ONL will play a larger role in the future as student diversity increases and attention spans shorten.

Section 2 describes how to meet the objectives of a modern networking laboratory course. It discusses how exercises can be constructed that gradually expose the student to higher levels of cognitive demands and how multiple system views can be used to enhance learning. Section 3 describes how guided exercises have been used to reinforce fundamental computer networking concepts. Section 4 describes how multiple perspectives improves student understanding of analytical formulae. Section 5 discusses some areas for improvement. Finally, Section 6 concludes with a discussion of future work.

* This work was supported by the National Science Foundation (ANI 023826, CNS 0551651, REC 0632580, REC 0632586).

2. A NETWORKING LAB COURSE

The fundamental instructional objectives of a networking laboratory are essentially the same as those proposed by Ernst [4] over 20 years ago in the context of general engineering education. When restated in the networking context, they are:

- Deepen student understanding of networking concepts
- Develop experimental computer science skills
- Increase experience with advanced networking technology

But what can be done to achieve these objectives in a meaningful manner? What makes a good computer networking exercise? This section examines these objectives and presents some thoughts on achieving them. The following two sections present our early experiences with using ONL in four networking courses at two universities which are committed to excellence in both research and education.

Some of the benefits of using ONL can be explained by examining the cognitive capabilities that can be exercised by networking laboratory assignments. Bloom's taxonomy [1] defines six domains of increasing levels of cognitive capability: knowledge, comprehension, application, analysis, synthesis, and evaluation. For example, the *knowledge category* might involve merely recalling information whereas the *evaluation category* might involve assessing competing theories.

Unfortunately, many students employ a trial-and-error, most-recent-concept approach to problem solving. Also, their repository of relevant facts is nearly empty, and they have not developed confidence in dealing with new technology. The organization and conduct of a good laboratory course needs to be sensitive to this situation.

ONL allows students to conduct network experiments starting with a small amount of information and then building their repository of network facts and relationships through exercises that are paced to require an increasing volume of information and levels of cognition.

ONL is built around a set of real networking hardware components (extensible routers), which connect a number of end systems in the users' experimental network [3]. Users interact with their network through the Remote Laboratory Interface (RLI), an intuitive graphical user interface that provides a wide range of configuration variables and measurement points, allowing a user to control system operation and observe the effects of system parameters on traffic. Measurements can be

displayed on real-time charts, allowing users to see what's happening "under the covers."

In order to start with a small cognitive load for students, the instructor can provide the class with a preplanned configuration and monitoring file that sets up the network and opens up monitoring charts; and then ask students to observe the effect of route changes to simple network traffic supplied by the *ping* command. Figure 1 shows the main RLI window and one of the route tables from using such a preplanned experiment file. The large circle with the numbers along the periphery is a single NSP (Network Services Platform) router with its eight ports. Seven hosts are attached to the router, three through a Gigabit Ethernet switch. Although there are hundreds of control knobs and observation points, the student only needs to know about very basic routing, link capacity, and basic monitoring to see the effects of route table changes that the students can make themselves.

Figure 2 shows an example of the traffic bandwidth going through output port 2 to host n1p2 during a period when students delete the route table entry used to forward traffic. The description of the bandwidth chart is part of the experiment file given to the student and specifies the parameters and the format of the display. The students can generate traffic by logging into an attached host and running a traffic command like *ping*. Students first observe that traffic appears on the bandwidth chart once per second (the default packet transmission interval of *ping*). Part of the exercise also asks students to predict the appearance of the chart from theory and to observe the bandwidth at other places along the intended packet path. This approach emphasizes the benefits to learning from reconciling inconsistencies between different representations and the need to view redundant data as a further verification of the observations.

Eventually, students are asked to construct their own configuration file which includes routing tables for a particular routing topology, thus exercising their understanding of IP addressing and the longest-prefix-match routing algorithm. This gradual approach to exercises allows the student to start quickly but places increasing demands on their understanding of basic concepts and cognitive abilities. Misunderstandings can be quickly resolved if the student is given strategies for recognizing and resolving mistakes, thus increasing their confidence in solving problems.

Finding the connections between different representations of the same phenomenon is an important cognitive ability [2]. The availability of distinct but related types of measurements (queue length, delay, packet loss rate) in ONL allows students to understand system behavior from multiple perspectives. Users can select from a wide range of different metrics (or can create

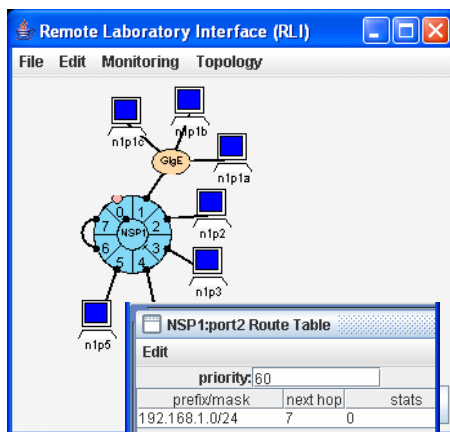


Figure 1. Route Table at Port 2.

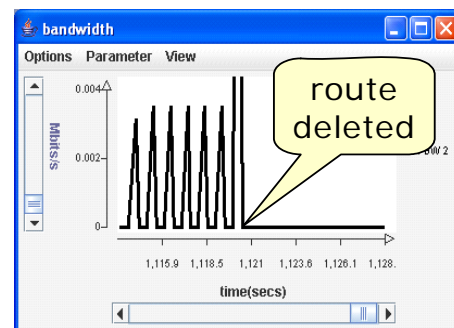


Figure 2. Ping Reply Traffic At Output Port 2.

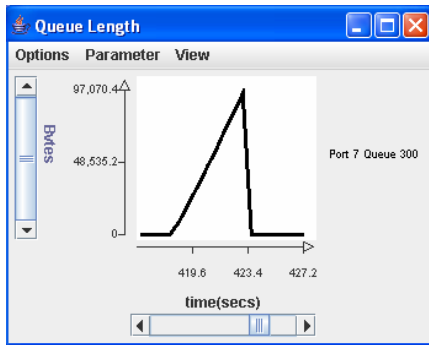


Figure 3. Queue Length (Port 7 Bottleneck).

their own metrics using plugins and/or applications running on the end systems) and can display these metrics directly on real-time charts.

Figure 3 shows the length of the bottleneck queue when constant rate UDP traffic is sent through the 8 Mbps bottleneck link at port 7 in Figure 1 at a rate slightly higher than its capacity. In this example, the student is asked to generate traffic at a rate that is about 1.87% faster than the capacity of the link at port 7 for five seconds. Students are asked to verify that the slope of the queue length curve agrees with a rough calculation.

Moreover, students can directly observe the queuing delay experienced by a packet sharing the bottleneck queue using the *ping* command. If students send *ping* packets through the same bottleneck link during the 5 second data transfer, they will discover that each *ping* packet experiences an ever increasing delay (e.g., 0, 39, 78, 117, 146 msec) before returning to 0 msec when the UDP traffic source stops. Students are asked to explain this apparent regularity based on theory.

In addition, advanced students can develop, experiment with, and evaluate new network services by adding router functionality through software plugins. This activity requires the highest degree and levels of cognition. We have used this approach in one of our advanced networking courses where students implemented a packet priority service in one exercise and used plugins in their final course project.

Laboratory experiences are also valuable for their potential to strengthen competence in the design and conduct of experiments, the analysis of data, and the selection of exploration trajectories. Improving student understanding of the operation of complex, dynamic systems also helps students to understand probabilistic rather than strictly deterministic system behavior. The ability to directly observe the effects of various parameters on system behavior, allows students to solidify their understanding of the underlying principles and can help them appreciate the difference between analytical predictions and measurements of real systems.

3. GUIDED LABORATORY EXERCISES

One of the co-authors taught a graduate networking course where conventional laboratory assignments that used socket programming and ns-2 simulations [8] were replaced with ONL exercises. This change made the laboratory portion of this course much more accessible to students with non-networking backgrounds because the amount of prior knowledge and skills required to start using ONL is much less than required in conventional laboratory assignments. This aspect of ONL was particularly important for this course because it is required for all doctoral students in the computer engineering program with no

prerequisite networking course. Typically, more than one-half of the 20 students had no prior experience with networking.

The course is structured to briefly introduce the basic networking protocol stack at the beginning of the semester and then expand into more advanced topics. The course takes a systems view of networking starting with individual links, then routers, and finally the entire network. The detailed topics that are covered in the course are:

- Introduction and the Internet protocol stack (5 lectures)
- Queuing systems (2 lectures)
- Router design and routing protocols (3 lectures)
- Router implementation issues (route lookups, packet classification, scheduling algorithms) (4 lectures)
- Network-level issues (congestion control, Internet topology, network measurement) (5 lectures)
- Other topics (security, multimedia, wireless) (5 lectures)

We use the Kurose and Ross textbook [5] augmented by a few seminal papers from high-quality networking journals and conferences.

The assignments for this course should ideally follow the same sequence of topics. However, since laboratory experiments use a real system, it is difficult to start at a very fundamental level. For example, it would be desirable to experiment with a single link and queue during the early part of the course when queuing systems are studied. But it is impossible without using end-systems and a small network to interconnect them. Therefore, it is not possible to have a perfect one-to-one mapping of topics covered in the lecture with topics covered in the lab. But ONL does allow us to hide many of the complexities by using default features.

In the Spring 2005 version of the course that did not use ONL, the laboratory assignments were:

- Lab 1: Implement an HTTP proxy. Evaluate system in terms of functionality and performance.
- Lab 2: Implement a reliable transport protocol on a channel. Evaluate functionality and performance for a range of reliability scenarios.
- Lab 3: Explore ns-2 simulator. Observe TCP and UDP congestion collapse.
- Lab 4: Analyse network trace. Extract packet-level flow-level statistics.

Labs 1 and 2 required students to program in C or JAVA using socket abstractions. Lab 3 required an understanding of the ns-2 software package. Lab 4 required the use of the *pcap* package in a custom C program.

Each assignment posed a significant challenge to students because, in effect, each exercise required students to construct a new laboratory tool (a program) using a large number of programming details before they could begin to explore the fundamental concepts of interest. This is in marked contrast to our recent use of ONL which allowed us to quickly setup simple exercises that would guide the students through incrementally more challenging problems.

In Spring 2006, we replaced the first three laboratory assignments with ones using ONL and kept the last one. The three new assignments were:

- Lab 1: Configure different network scenarios. Measure simple statistics to track functionality and performance.

- Lab 2: Analyse queueing effects. Explore queueing behavior during TCP transfer. Analyse relationship between queue length and delay.
- Lab 3: Analyse TCP behavior. Explore competition between TCP and UDP flows. Compare results from ONL with those from ns-2 simulation. Analysis of TCP congestion control behavior for different buffer sizes.

Two of the differences reflected in the first three exercises in this new approach is the focus on more fundamental concepts such as routing and queueing and on incrementally developing facility in constructing knowledge relationships through actual and thought experiments.

Some of the benefits of using ONL can be explained by comparing the two approaches with respect to the six cognitive objectives in Bloom's taxonomy. Table 1 shows the main cognitive skill (☑) and related skills (✓) employed in the above exercises for both approaches. By using ONL, students can start in application (Ap) and analysis (An) modes before moving on to synthesis (Sy) and evaluation (Ev), thus providing better pacing of cognitive requirements.

Anecdotal evidence from informal student surveys indicated a high level of enthusiasm because of the hands-on experience and the connection between theory and practice. A surprising result was the citation of ease-of-use since the students got all of their information on ONL through the Web tutorial pages. In an advanced networking course, a special FAQ was maintained by ONL developers to assist students with router plugin projects.

4. MULTIPLE REPRESENTATIONS

Research in learning has demonstrated the importance of predicting experimental outcomes and checking that results from multiple representations such as graphs, tables, formulae are consistent. ONL already has charting facilities that often allow students to view behavior from multiple perspectives. This section demonstrates how ONL charting and data facilities aid in the understanding of pipelined file transfer.

Kurose and Ross [5] advocate an analytical approach to teaching computer networks where the performance of designs is expressed in terms of packet transmission, propagation delay, and queueing delays. Most students understand that the end-to-end delivery time of one packet of length L over an empty path with n links is given by:

$$T(1) = \sum_{i=1}^n [d(i) + L/c(i)]$$

where $d(i)$ is the propagation delay, and $c(i)$ is the capacity of link i . On the other hand, many students struggle to derive correct analytic expressions for more complex settings (e.g., the sender transmits w packets back-to-back and there is sufficient network buffering to avoid packet drops). A common mistake is to express the end-to-end delivery time of w packets as $wT(1)$. They miss the main point that the packets form a pipeline bottlenecked at the lowest capacity link. The correct expression that accounts for the pipelining effect is:

$$T(w) = T(1) + (w-1)L/c(\min)$$

where $c(\min)$ is the capacity of the slowest link. This form captures the idea that the first packet takes $T(1)$ to travel from the sender to the receiver, and then each of the remaining $(w-1)$ packets arrives at the receiver $L/c(\min)$ after the preceding packet.

The effect of pipelining can be easily explored using ONL using a setup similar to the one shown in Figure 1. Packets from host n1p2 enter port 2 where they are forwarded to port 7 and then out the bottleneck link to port 6 where they are forwarded to port 3 and then out to host n1p3. The bottleneck link at port 7 is still 8 Mbps. But the capacity of queue 300 is increased to 10 MB, enough to handle a single file transfer of 10 MB. A 50 msec delay plugin is inserted at input port 2, and the return path is configured with negligible delay.

The effects of pipelining can be demonstrated by running two experiments where we send 6,805 packets each containing 1,470 bytes of data:

- *Non-Pipelined:* A packet is sent only after the preceding packet reaches the receiver (n1p3).
- *Pipelined:* All packets are sent back-to-back at a rate substantially higher than the bottleneck link capacity of 8 Mbps.

Students observe the queue length at and the output bandwidth of the bottleneck in both cases and are asked to explain the different results including the slope and peak value of the queue length plots and the peak value and duration of the bandwidth plots.

Figure 4 shows the beginning period of the bandwidth coming out of the bottleneck link. The bandwidth is about 240 Kbps, substantially lower than the 8 Mbps capacity of the link and lasts for a little over 340 seconds. The queue length chart (not shown), indicates no packet backlog. The bandwidth and queue length observations can be explained by realizing that only about 20 packets per second can be delivered through a 50 msec delay path

Course	Lab	Cognitive Objective					
		Kn	Co	Ap	An	Sy	Ev
Pre-ONL	1	✓	✓	✓	✓	☑	✓
	2	✓	✓	✓	✓	☑	✓
	3	✓	✓	✓	✓	☑	
	4	✓	✓	✓	☑	✓	
ONL	1	✓	✓	☑			
	2	✓	✓	✓	☑		
	3	✓	✓	✓	☑		✓
	4	✓	✓	✓	☑	✓	

Table 1. Traditional Versus ONL Exercises.

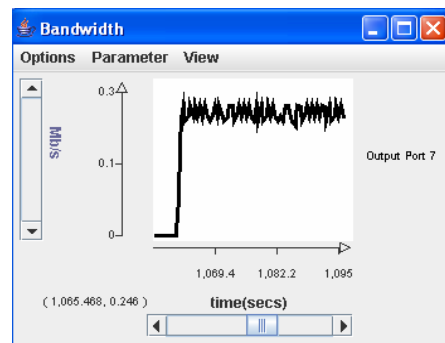


Figure 4. Port 7 Output Bandwidth (Non-Pipelined).

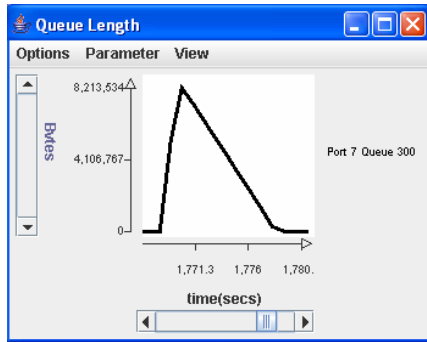


Figure 5. Bottleneck Queue Length (Pipelined).

with no pipelining and that each packet travels between the sender and the receiver alone. A rough calculation assuming 12,000-bit packets (1,500 bytes) leads to an estimate of 240 Kbps. Furthermore, this 20 packet per second rate can be directly observed at a number of different points in the network through ONL's extensive set of measurement points.

Figure 5 shows the size of the backlog at the bottleneck queue when the UDP traffic is sent at 80 Mbps. The queue length chart shows that the queue is backlogged for about 10 seconds which is the time required to transmit 80 Mbits at 8 Mbps. Furthermore, this 10 seconds is also the delivery time $T(w=6,805)$ when the delivery time is dominated by the transmission term $wL/c(\text{min})$. The bandwidth chart (not shown) verifies this transmission time period which is substantially less than the 340 seconds in the non-pipelined case.

This simple exercise can be extended in a number of ways. An even greater understanding of the effect of pipelining can be gained by viewing together multiple representations such as charts of analytic formulae, data transfer progress, and space-time event chart (i.e., when a packet is sent, received, and acknowledged) and checking for consistency among the different views. Pre-written basic UDP programs that allow the user to vary the window size w of packets that can be concurrently in flight can be instrumented to collect packet arrival and departure event times at the millisecond time scale. This data can then be replayed to generate a space-time diagram showing the concurrency occurring in the pipelining case. A more advanced form of this exercise is to have students extend the pre-written UDP programs to include logic for detecting and responding to packet drops which can be emulated through an ONL-supplied plugin that both delays and probabilistically drops packets.

5. IMPROVEMENTS

Four networking courses (undergraduate and graduate) have used ONL exercises over the past year. Student surveys and instructor interviews lead us to the following observations:

- Engineering students (even non-networking ones) are enthusiastic and remain so if exercises are properly paced and integrated with the course material.
- Students find the system easy-to-use but consult the on-line tutorial pages only when encountering problems.
- Instructors can develop their own exercises with little effort by using the growing on-line repository of exercises at the ONL website.

But to be successful, particular attention still needs to be paid to course conduct: pacing, organization, and insuring student

confusion is addressed promptly. A tool that recorded laboratory sessions for asynchronous review by a tutor or automated analysis might help in identifying conceptual misunderstandings.

To have a broader impact on laboratory courses in general, a better approach to assessment is needed. Assessment, here, refers to collecting data that measures student learning and the effectiveness of particular pedagogies. We need to measure the extent to which students can recognize relationships through the interpretation of data provided through real-time charts. We need to identify how students form, reinforce and/or modify their "mental models" of the underlying system and determine the extent to which this allows them to improve their ability to predict system behavior under conditions that they have not experimentally explored. Tools such as *agar* [6] that collect data on student solutions might provide raw data that could begin to be used in addressing learning research questions.

6. CONCLUSIONS

We have described how a virtual network laboratory can meet the objectives of a modern networking laboratory course. We have discussed how exercises can be constructed to provide a good learning experience through a structured exposure to networking concepts using the multiple perspectives provided by ONL. This is in contrast to a more traditional approach where socket programming is used. Although we have identified some useful approaches, much work remains in understanding and documenting the learning experience.

There are plans to include Network Processor platforms that will provide both programming flexibility and high performance. This will include the implementation of the high-speed forwarding part of a complete IP router with support for packet classification, per flow queuing, flexible statistics collection and a plugin subsystem for user-specific services. Furthermore, the source code will be made available to students thus allowing them to make custom modifications to the new routers. The above enhancements will improve the educational experience by giving students an even higher degree of experimental control.

7. REFERENCES

- [1] Benjamin S. Bloom, et. al. (Ed.), "Taxonomy of Educational Objectives," New York: David McKay Company Inc, c1956-1964.
- [2] J. Confrey and A. Maloney (in press), "A Theory of Mathematical Modeling in Technological Settings," in Applications and Modeling in Mathematics Education, edited by W. Blum and H. Henn, Springer, 2006.
- [3] John D. DeHart, et. al., "The Open Network Laboratory," *Proc. SIGCSE*, March 2006.
- [4] E. W. Ernst, "A New Role for the Undergraduate Engineering Laboratory," *IEEE Trans. On Education*, Vol. E-26, No. 2, May 1983, pp. 49-51.
- [5] James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet* (3rd Ed.), Addison Wesley, 2004.
- [6] Titus Winters and Tom Payne, *What Do Students Know?*, Proc. First Intl. Computing Education Research Workshop (ICER05), 2005.
- [7] Open Network Laboratory, <http://www.arl.wustl.edu>.
- [8] The Network Simulator ns-2, <http://www.isi.edu/nsnam/ns/>