An Educational Networking Framework for Full Layer Implementation and Testing

Keunhong Lee, Joongi Kim, Sue Moon
Department of Compute Science, KAIST
{khlee, joongi}@an.kaist.ac.kr, sbmoon@kaist.edu

Categories and Subject Descriptors

C.2.2 [**Conputer-Communication Networks**]: Network Protocols—*Protocol verification*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing tools*; K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education, self-assessment*

General Terms

Design, Verification

Keywords

Educational Networking Framework, Full Layer Implementation, Automated Test Suite, Network protocols, TCP, IP

1 Introduction

In our computer science discipline, hands-on projects challenge students to build systems they learn in class. These projects have always been an integral part of our curricula. Kurose and Ross supplement their textbook on computer networking with Wireshark[1] labs and programming assignments for network application[3]. Wireshark labs help students to grasp quickly the workings of today's Internet via packet trace analysis without writing code. MYSOCK and STCP[2] have been used in undergrad networking courses for students to implement the socket API and a simplified TCP connection mechanism to work over simulated packet losses and reordering. Advanced graduate-level courses often include experiments with the network infrastructure. The popular network simulation tool, `ns2`[3], has accumulated an extensive set of protocol implementations in the past two decades, but the goal of `ns2` is to examine the protocol performance against other competing traffic. While `ns2` operates strictly in a simulated environment, emulab[4] and ONL (Open Network Laboratory) [2] offer an emulated networking environment with ease of access and configurability. VNS [1] and its successor Mininet [4] have empowered researchers with a container-based emulation environment that closely matches the performance of a real testbed with high fidelity.

Among the multitudes of protocol and networking tools, there lacks an environment where students can implement and test the full protocol layer in a realistic setting. In this poster we present ELSA (Educational Layered System with Auto-testing), a framework for students to implement TCP, IP, and routing protocols, and test against adversary implementations. We have designed ELSA, implemented it and are currently using it in CS341, an undergraduate computer networking course at KAIST. In the next section we present the design and main features of ELSA.

2 Design and Key Features

We have designed and implemented ELSA with the following educational goals in mind.

2.1 Complete experience on layered abstraction

To offer a complete experience on implementation of a layer in networking stacks, programming assignments should cover all of the following elements:

- demultiplexing calls from the lower layer (TCP)
- multiplexing calls from the upper layer (TCP)
- context management (IP and TCP)
- functions in protocol specification (IP and TCP)

We have designed the following hands-on projects to address the above elements:

- In the TCP layer, students are to implement connection establishment and tear-down. ELSA supports both reliable and unreliable modes for testing data transmission. Congestion and flow control of TCP is also part of the assignment. The challenge here is to manage multiple connections and their states in a single layer.
- The IP layer generates pcap-compatible logs[5] with the actual Ethernet packets instead of simulated data streams, which allows debugging and inspection via packet tracing tools such as Wireshark. This layer covers topics on route lookup, IP switching, persistent connection tests with the TCP layer and routing protocols (e.g., distance vector) under link failures.

[1] `http://www.wireshark.org/`
[2] `http://www.stanford.edu/class/cs244a/hw3/hw3.html`
[3] `http://www.isi.edu/nsnam/ns/`
[4] `http://www.emulab.net`

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631467.

[5] MIME type *vnd.tcpdump.pcap*. See `http://www.iana.org/assignments/media-types/application/vnd.tcpdump.pcap`

2.2 Complete experience over a network of hosts

The data link layer is implemented using Linux UDP sockets and a single host is realized as a single process. Students can build a network of hosts by having multiple ELSA processes either on a single machine or over a distributed environment. By replacing the virtual driver that implements the data link layer with a real user-level Ethernet device driver, we could turn ELSA into a barebone Linux networking stack. A network of four nodes connected in tandem is given to students as a reference network configuration.

3 Improvements from Experience

We have collected feedback from students and teaching assistants at the end of every semester when this framework was used. Here we summarize the comments, and describe our solutions in response to them.

Comments from students:

- **It was difficult to develop and test only one side of a coupled function.** Networking primitives are coupled with their counterparts. For example, the completion of connect() requires a correct implementation of accept() on the adversarial side. Students could not verify correctness of a partial, one-sided implementation.
- **Framework internals were not hidden from students.** Students had to understand the internal behaviours of the framework to write correctly working code. In particular, previous versions have exposed internal data structures for inter-layer communication. For example, students had to copy and manage underlying pipe file descriptors used by the framework when accept()-ing a new connection.

Comments from TAs:

- **Setting up a testing environment was time-consuming.** Setting up a multi-process virtual environment to grade an assignment was tedious, time-consuming, and not straightforward to automate.
- **Matching logs to the correct behavior was challenging.** Every student had to demonstrate the submitted code in person to the TA and ran through the log line by line with the TA in order to verify that the code was correct. It was almost impossible to find a needle in a haystack; investigating more than 1K lines of TCP implementation (e.g., 1187 LoC in a reference implementation[6]) was impractical when grading tens of students. Our practice was to check console outputs (logs) and let students explain what was going on. It was time-consuming and difficult to prove incorrectness when TAs were faced with verbose output.

Our solutions:

- **Combine multiple instances of function implementations:** ELSA enables incremental tests of coupled functions by combining multiple implementations of a single layer (e.g., TCP) in a single process. For example, ELSA combines both students' version of the TCP layer and a reference implementation in a single process for testing, which allows incremental and independent tests of every function.
- **Make the framework event-driven:** We reduce the amount of code to understand when beginning each assignment by adapting an event-driven structure. Students can receive events by implementing proper callback functions instead of polling events from the framework. This approach reduces the amount of skeleton code (we provide 84 lines of skeleton and 48 lines for header files). The skeleton resembles that of the linux driver code. As all the events are totally ordered, students can focus on the core functions rather than complicated synchronization issues.
- **Build an automated test suite for evaluation:** In order to reduce the effort that goes into grading, we have developed a complete test suite for all possible code execution paths in the TCP state transition diagram for network events such as packet reordering and losses. The test suite outputs grading results from a single issue of a command. Albeit it is almost an independent part of the framework, the test suit is the most time savior in our grading effort.

[6]The number of lines are measured with CLOC. See `http://cloc.sourceforge.net/`

4 Summary and Future Work

ELSA is an educational framework for transport and network layer implementation. Our improvements are based on eight years of accumulated lab expreiences. In Spring 2014 we are already benefitting from the improvements. We could cut the number of TAs by half for the class size 50% larger than last year. At the end of the semester we will release the framework as an open-source software and encourage other universities to try ELSA.

We have following plans for ELSA. First, we will extend our improvements to the IP layer. The current version of ELSA has reference implementations of the IP layer including fragmentation and reassembly, but lacks the automated test suite. Second, we are considering migrating ELSA's data link layer to a more general platform, such as Mininet or native Linux kernel. Third, we should extend the framework to include IPv6. With the address exhaustion of IPv4, IPv6 is gaining ground and educational networking systems should reflect this change.

5 Acknowledgments

ELSA is based on KENS (KAIST Educational Network System) developed by Junehwa Song and his students. The original motivation and design of KENS has been a great inspiration and we are greatly indebted. This work was supported by the Basic Science Research Program by the National Research Foundation of Korea (NRF) of MSIP (2014R1A2A1A01007580).

6 References

[1] M. Casado and N. McKeown. The virtual network system. In *ACM SIGCSE Bulletin*. ACM, 2005.

[2] J. DeHart, et al. The open network laboratory. In *ACM SIGCSE Bulletin*. ACM, 2006.

[3] J. F. Kurose and K. W. Ross. *Computer networking: a top-down approach featuring the Internet*. Pearson Education India, 2005.

[4] B. Lantz, et al. A network in a laptop: rapid prototyping for software-defined networks. In *ACM SIGCOMM HotNets*. ACM, 2010.