Open Educational Resources for Computer Networking

O. Bonaventure
UCLouvain, Belgium
olivier.bonaventure@uclouvain.be

Q. De Coninck
UCLouvain, Belgium
quentin.deconinck@uclouvain.be

F. Duchêne
UCLouvain, Belgium
fabien.duchene@uclouvain.be

A. Gégo
UCLouvain, Belgium
anthony.gego@uclouvain.be

M. Jadin
UCLouvain, Belgium
mathieu.jadin@uclouvain.be

F. Michel
UCLouvain, Belgium
francois.michel@uclouvain.be

M. Piraux
UCLouvain, Belgium
maxime.piraux@uclouvain.be

C. Poncin
UCLouvain, Belgium
chantal.poncin@uclouvain.be

O. Tilmans
Nokia Bell Labs, Belgium
olivier.tilmans@nokia-bell-labs.com

ABSTRACT

To reflect the importance of network technologies, networking courses are now part of the core materials of Computer Science degrees. We report our experience in jointly developing an open-source ebook for the introductory course, and a series of open educational resources for both the introductory and advanced networking courses. These ensure students actively engage with the course materials, through a hands-on approach; and scale to the larger classrooms and limited teaching staff, by leveraging open-source resources and an automated grading platform to provide feedback. We evaluate the impact of these pedagogical innovations by surveying the students, who indicated that these were helpful for them to master the course materials.

CCS CONCEPTS

• **Applied computing → Education**; • **Networks → Network protocols**; **Programming interfaces**; **Network experimentation**;

KEYWORDS

Education, Open Educational Resources

1 INTRODUCTION

In less than half a century, computer networks have revolutionized our society. While a few packets were exchanged between ARPANet nodes almost fifty years ago, nearly anyone can now access the Internet at any time and anywhere using mobile devices. The structure of networking courses is closely linked to the popularity of the technology. The first courses were aimed at training the researchers who were developing the new networks and protocols, e.g., using Bertsekas and Gallager's textbook [2]. As the popularity of the Internet rose, a growing number of universities included networking courses in their curriculum in the eighties and nineties. As the students were not exposed to real computer networks, many textbooks adopted a bottom-up approach starting from the physical layer and moving progressively up to the application layer, such as Andrew Tanenbaum's textbook [16]. Since the late 1990s, most students have been exposed to the Internet. For such students, and even more for today's ones, a top-down approach is much more motivating. This approach has been used by Jim Kurose and Keith Ross [10]. They start from the application layer by leveraging the practical knowledge of students. Then, they dissect the lower layers and reveal the main principles and protocols used on the Internet.

During the last ten years, we have written and expanded a free and open-source networking textbook [3] that has been adopted by various universities around the world to teach the introductory networking course. The first edition of our open-source networking textbook used the top-down approach. After a few years, feedback from the students indicated that they had difficulties with this approach. When a layer is explained using this approach, students need to learn the basic principles, algorithms and details of the deployed protocols all at the same time. Many of them had difficulties to separate key principles from protocol details. It has been demonstrated that students achieve a deeper understanding of a topic both by frequent exercises and by effective feedback [5]. However, due to the sharp increase of students attending the course, we also faced the challenge of a shrinking amount of time that professors and teaching assistants can dedicate to a given student. To address these issues, we revised the course organization and adopted a hybrid approach.

The first half of the course introduces all the key principles of computer networking using a bottom-up approach. It begins with a brief description of the physical layer, then describes the datalink layer using an abstract protocol to illustrate the principles of reliable delivery protocols. It then introduces the different organizations of the network layer, the separation between the control-plane and the data-plane, as well as key routing algorithms such as link-state and distance vectors without diving into protocol specific issues. The course then explores the transport layer, and explains the need for congestion control. Finally, key principles for network security conclude the first half of the course.

The second part of the course adopts a top-down approach using the key Internet protocols to illustrate how the principles described earlier are put into practice in the global Internet. Students learn practical aspects of employing those key Internet protocols through a combination of projects, labs and web-based tools. The automation of practical exercises helped us to scale up the course with the growing number of participants. In particular, we leverage an automated grading platform named INGInious [6] to provide detailed feedback on student answers, which improve their motivation and understanding of the course [1, 5].

In this paper, we report on our experience in developing various teaching materials to supplement networking courses. This paper is organized as follows. In Section 2, we report on our experience with a set of online open educational resources enabling students to better understand various introductory networking concepts. In Section 3, we present three projects enabling students to better understand how networking protocols are implemented and deployed. We also report on the lessons we learned when using these projects. To quantify the impact of these pedagogical innovations, we sent a survey to students who attended our introductory course. We received 128 replies, mainly from last year's attendees, but also from students who took the course during the previous two years. We provide the key findings of this survey throughout the text. Finally, we conclude with a discussion of the benefits of open educational resources.

2 TEXTBOOK VERSUS INTERACTIVE EBOOK

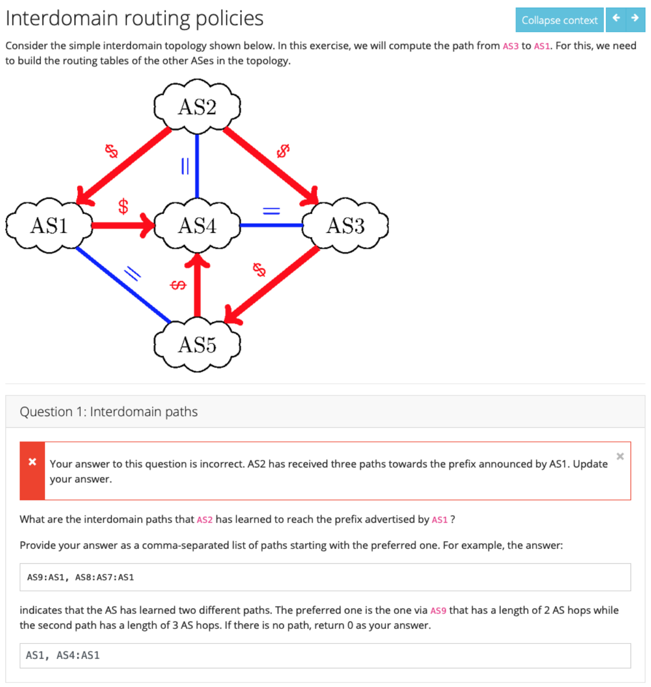
The first version of our ebook, written a decade ago, was structured as a regular book that was distributed online in PDF and HTML format. Many students printed the ebook and annotated it. As indicated by various studies [14], students tend to learn better on printed books than purely online. However, our experience shows that an online approach brings several benefits to a networking ebook. Our first modification was to add hyperlinks to all RFCs and articles cited in the bibliography. This was a simple change, but shortly after we were positively surprised that a growing fraction of students asked questions about some of these references and cited them correctly in their project reports. 55% of the surveyed students confirm that they follow (always or frequently) links to find additional information.

Students not only learn by reading the teaching material in the ebook and by listening to their professors or teaching assistants. They also learn by answering questions. An important benefit of having an online ebook is that it enables the students to interact with the course in different ways. Throughout the years, we have developed several types of exercises enabling students to test their knowledge of the teaching material during their learning process. When answering to these exercises, students often make mistakes which are integral part of their learning process. Students can benefit from these mistakes provided that they receive accurate feedback rapidly.

We integrated the ebook with the INGInious code-grading platform [6]. INGInious is an open-source project enabling professors to propose various types of exercises to students, and to automatically provide feedback and grade their answers. Most INGInious exercises require the students to write code in a programming language. This code is then compiled and graded by unit tests that verify whether the student correctly answered the question. INGInious exercises are implemented as scripts that receive answers provided by a student and compute the desired feedback. As such, INGInious also supports open questions (e.g., where the answers are short numerical values) and multiple-choice questions.

2.1 Multiple choice and short questions

Our ebook includes dozens of interactive multiple choice questions covering a wide range of topics. In order to encourage the students to answer these questions several times (e.g., during their first read, and later when they prepare the exam), each question contains two sets of answers: positive and negative ones. When displayed, each question is then dynamically constructed by a JavaScript module that randomly selects *one* correct answer and *n* invalid ones. To provide feedback, each possible answer has an associated comment. As the student selected their answer, the corresponding comment is then shown. For invalid answers, the comment thus provides the explanation that enables the student to correct their misunderstanding. 58% of our students report they answered (always or frequently) the multiple choice questions directly while reading the ebook, and 88% during the semester. 73% reused them while preparing for the exam.



Interdomain routing policies

Consider the simple interdomain topology shown below. In this exercise, we will compute the path from AS3 to AS1. For this, we need to build the routing tables of the other ASes in the topology.

Question 1: Interdomain paths

Your answer to this question is incorrect. AS2 has received three paths towards the prefix announced by AS1. Update your answer.

What are the interdomain paths that AS2 has learned to reach the prefix advertised by AS1 ?

Provide your answer as a comma-separated list of paths starting with the preferred one. For example, the answer:

```
AS9:AS1, AS8:AS7:AS1
```

indicates that the AS has learned two different paths. The preferred one is the one via AS9 that has a length of 2 AS hops while the second path has a length of 3 AS hops. If there is no path, return 0 as your answer.

```
AS1, AS4:AS1
```

Figure 1: Questions with a short answer provide immediate feedback to the students.

Multiple choice questions are not a panacea. For some concepts, it is interesting to ask questions requiring a short answer such as a number or a few words, prone to automated grading through pattern matching. It is often more challenging for a student to answer such a question than to guess one answer among *n* choices. The expected answer structure (i.e., pattern) lets us still use INGInious to automate the grading and feedback generation. One example is shown in Figure 1. This exercise is used while the students learn the standard BGP routing policies [7]: *customer-provider* (represented as a directed red arrow) and the *shared-cost peerings* (represented as a blue line with the equal sign). The students try to determine the BGP routes that each AS receives for a given prefix and enter the corresponding AS paths. INGInious then parses the answer, and immediately provides detailed feedback enabling the students to learn from their mistakes if any.

2.2 Socket programming

Most networking courses include an introduction to socket programming. This low-level API is the classical way to develop networked applications that interact with the transport layer, mainly TCP and UDP. Several textbooks [8, 15] describe these interactions in detail. Despite the qualities of these textbooks, many students have difficulties in understanding several of the key concepts that underlie the development of networked applications. For example, some students have difficulties in understanding the difference between big-endian and little-endian. Many of them forget to check the return values of system calls or cannot correctly parse data received from the network.

As an example, let us illustrate one of the INGInious questions that verifies whether a student correctly uses the socket layer to interact with UDP. The students need to implement a client that sends a vector of integer in a UDP datagram to a server and receives in response a UDP datagram with the sum of these integers. Both the client and the server are implemented in one function whose correct operation is verified by INGInious. These questions are verified with a script that compiles the C code and then runs a series of unit tests. These unit tests validate different aspects of the code written by students. Some of these tests include simple assertions verifying that the correct value is returned for a given set of parameters. Others are more subtle. For example, to verify that the students correctly check the return values of the memory allocation functions, our tests use library wrappers to intercept these functions and force them to return errors in some tests. We do the same for system calls such as `send` and `recv`. Our tests also verify that the students use the network byte order when sending and receiving integers.

We developed a similar exercise that uses TCP instead of UDP. In this case, the client opens a connection to the server, sends its vector of integers and waits for the answer. Our tests verify that the client and server codes correctly use the socket interface. Some of our tests cover corner cases that students often ignore at a first glance. For this exercise, many students expect that when a server issues a `recv` system call, it will also return the entire vector of integers. This is what they usually observe from a small test in the LAN on their computer. In reality, the `recv` system call returns the number of bytes that have been received in-sequence. If the vector was sent in multiple packets on a WAN, only a fraction of them could be available when the server calls `recv`. Some of our tests simulate this behavior to encourage the students to immediately write code that handles all possible cases.

2.3 Learning protocols by dissecting packets

One of the objectives of many networking courses is to enable the students to understand how network protocols operate. Most textbooks include a textual description of the protocol which includes the packet format, a finite state machine and some examples. This textual description is fine for the assiduous students, but many protocols rely on conventions that need to be well understood. While these conventions can be explained with text or with some pseudo-code, students better learn about these conventions by observing and interacting with the protocols. For many students, using a packet dissector like `Wireshark` or `tcpdump` makes the network protocols become real, and they learn by observing the different packets that are exchanged inside a network.

When observing packets, each of the fields that are included in their headers have a specific role. Some of them have different roles that depend on the values of specific flags. To reinforce their understanding, our ebook includes a series of INGInious exercises that require the students to predict the value of specific fields of packets in a given exchange. To develop such exercises, a professor simply needs to collect packet traces in a real or emulated network.We extended INGInious with a simplified packet dissector [12] that presents packet traces similarly to `Wireshark`. The student can look at the content of each packet and observe the values of the different fields. The teacher can then mask some fields of the packet header and ask the student to predict their values based on the context provided by the other packets in the trace.

A simple example is provided in Figure 2. More precisely, INGInious shows the three packets that compose a TCP handshake in random order. Packet `#2` is the `SYN` packet sent by the client. Packet `#1` is the `SYN+ACK` that was returned by the server and packet `#0` is the third packet of the handshake. In this exercise, INGInious randomizes the packet trace and the students have to reorder them. As the student submitted a wrong solution, Figure 2 thus additionally shows the feedback provided by INGInious, when grading the exercice.

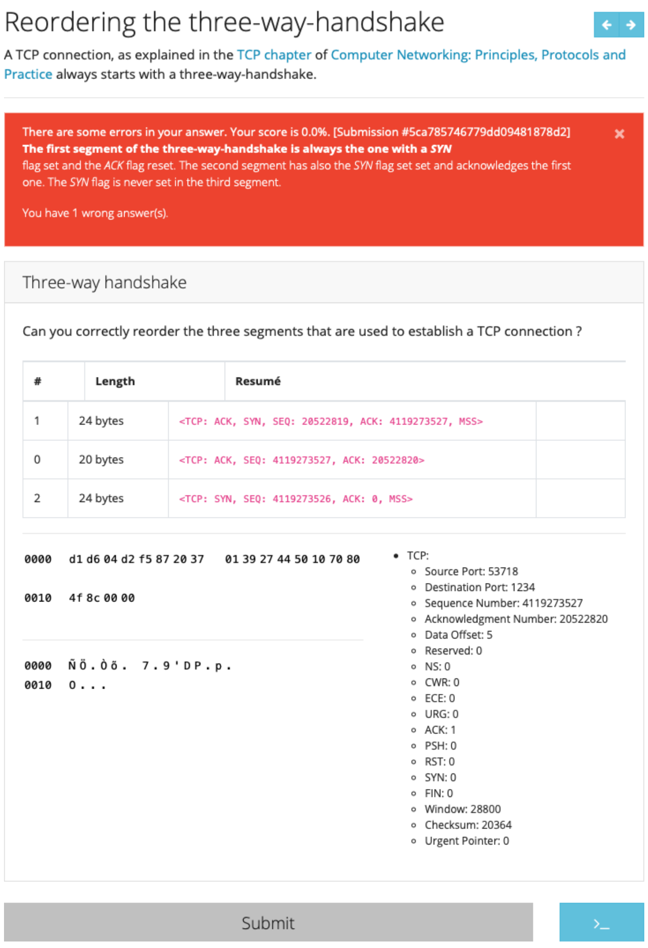
According to our survey, 66% of the students who used this exercise found it useful to understand the 3-way handshake.

2.4 Experimenting with real implementations

The INGInious exercises described in the previous sections enable the students to learn the basic principles of the Internet protocols. However, this is not sufficient to let them grasp how the protocols behave in real networks, e.g., due to their distributed nature, or due to implementation-specific behaviors. We thus complement the INGInious exercises with hands-on exercises using real protocol implementations.

Running virtual labs Many universities and enterprises use labs equipped with routers, switches servers, and many cables. However, given the growth in the number of Computer Science students, it became impossible for us to manage and assign such physical labs for more than two hundred students.

Fortunately, it is possible to leverage open-source frameworks, such as Netkit [13] and Mininet [11], to develop virtual labs that can be easily used by students. In particular, Mininet leverages the Linux namespaces to efficiently emulate several hosts, switches and routers on a single Linux host. This enables the distribution of virtual machines to the students, in which they can instantiate the switches and hosts for their virtual lab. Mininet's original purpose is to enable experimentation in Software-Defined Networks (e.g., OpenFlow). While it is possible to add routing daemons to Mininet and configure them manually, many students lack the system administration background required to efficiently configure such daemons. This creates a barrier preventing them from experimenting easily with real routing protocols. To cope with this problem, we have developed a set of extensions to Mininet abstracting the configuration and management of different routing daemons.



Reordering the three-way-handshake

A TCP connection, as explained in the TCP chapter of Computer Networking: Principles, Protocols and Practice always starts with a three-way-handshake.

There are some errors in your answer. Your score is 0.0%. [Submission #5ca785746779dd09481878d2]
The first segment of the three-way-handshake is always the one with a *SYN*
flag set and the *ACK* flag reset. The second segment has also the *SYN* flag set set and acknowledges the first one. The *SYN* flag is never set in the third segment.

You have 1 wrong answer(s).

Three-way handshake

Can you correctly reorder the three segments that are used to establish a TCP connection ?

#	Length	Resumé
1	24 bytes	`<TCP: ACK, SYN, SEQ: 20522819, ACK: 4119273527, MSS>`
0	20 bytes	`<TCP: ACK, SEQ: 4119273527, ACK: 20522820>`
2	24 bytes	`<TCP: SYN, SEQ: 4119273526, ACK: 0, MSS>`

```
0000   d1 d6 04 d2 f5 87 20 37   01 39 27 44 50 10 70 80

0010   4f 8c 00 00
```

```
0000   Ñ Ö . Ò õ .   7 . 9 ' D P . p .
0010   O . . .
```

- TCP:
  - Source Port: 53718
  - Destination Port: 1234
  - Sequence Number: 4119273527
  - Acknowledgment Number: 20522820
  - Data Offset: 5
  - Reserved: 0
  - NS: 0
  - CWR: 0
  - ECE: 0
  - URG: 0
  - ACK: 1
  - PSH: 0
  - RST: 0
  - SYN: 0
  - FIN: 0
  - Window: 28800
  - Checksum: 20364
  - Urgent Pointer: 0

Figure 2: Students learn the TCP three-way handshake by reordering packets.

`IPMininet` [18] is a set of Python classes extending Mininet which provides a declarative API to instantiate IP networks. It enables professors to distribute network topologies with specific routing configuration, that student can then experiment with. For example, we use `IPMininet` to deploy static routes on a network, where some destinations are unreachable due to incomplete forwarding tables or forwarding loops. Students are then asked to correct the problem. For OSPF and RIP, `IPMininet` provides abstractions so that a simple Python script can configure the IP prefixes, set the link weights and launch the routing daemons. The students can then capture packets, observe the routing tables of the virtual routers and use `ping` or `traceroute` to interact with the network. The BGP API of `IPMininet` is more comprehensive. With a few lines of Python, students can configure simple BGP routers that advertise prefixes. We also provide abstractions that create the routing policies required to support *customer-provider* and *shared-cost peerings*. `IPMininet` also supports the creation of iBGP sessions and the configuration of Route Reflectors. This enables the students to explore how BGP is used in large Internet Service Provider networks. The list of network services for which `IPMininet` provides a declarative API goes well beyond the above routing daemons, and also includes e.g., PIM, BIND, SSH, IPTables, Radvd—the complete list of which is available on its repository [18].

Unit-testing protocols To encourage students to explore TCP and understand how mature TCP implementations such as the Linux TCP stack behave, we use `packetdrill`. Packetdrill [4] is a unit-testing tool that allows verifying the conformance of a TCP stack to some tests. It was designed to validate the Linux TCP stack. With `packetdrill`, students issue system calls such as `read` and `write`, inject TCP or even ICMP packets in the Linux stack with precise timing information, and observe how it reacts. By writing small `packetdrill` tests, students gain a more in-depth understanding of the operation of the Linux TCP stack, as well as the behavior of the available configuration knobs available in the implementation.

3 STUDENT PROJECTS

Students learn a lot through projects, which enable them to be creative and expand their understanding of the course topics. The main difficulty in creating student projects is to find the right balance between the time that the students spend on the project and the skills that they learn. We developed two such projects for the introductory networking course, spanning both of its halves. Put together, those projects are worth 35% of the networking course grade. In the first project, covering the basic principles, students implement a simple reliable transport protocol using the socket layer. This project is worth 20% of the course grade. During the second project, they analyze the architecture and performance of a popular web service, thus exploring how protocols are used in practice. This is worth 15% of the course grade. Finally, our advanced networking course is structured around the third project described in this section: the design and implementation an enterprise network. This third project shares some similarities with the mini-Internet project from Holterback *et al* [9]. The main difference is that we put emphasis on the services provided by enterprises that run on their network, such as DNS, DHCP and web servers, whereas the mini-Internet project is mainly focused on network operator aspects. For reference, both networking courses in which these project take place are worth 5 ECTS each, with 1 ECTS being equal to 25-30 hours of work – including course, practical sessions, personal study time, and evaluations.

3.1 Implementing a simple transport protocol

Our first project is proposed to pairs of students. This is a good match for the implementation of a client-server protocol, as one student can focus on the client side while the other focuses on the server side. During the project, students need to develop a simple transport protocol that uses window-based flow control, acknowledgments, and two retransmission techniques. We vary some details of the protocol every year to prevent students from simply reusing code written by their predecessors.

Description. The project is organized in four phases and the students benefit from each of them. The first phase is a one week-long bootstrap phase. During this phase, the students learn the basic principles of the reliable protocols and write small functions in C that are tested by INGInious. At the end of this phase, the students should have learned the basics of socket programming. They then develop their first implementation of the simple transport protocol during the three next weeks. This protocol runs on top of UDP for practical reasons. We provide them with a Wireshark dissector to analyze packets and a simple link simulator that introduces delays, reordering and losses on a given UDP flow. At the end of this phase, each pair of students has developed a prototype implementation of the simple transport protocol. During the third phase, spanning a week, we organize interoperability tests in which each student pair is required to test its implementation with at least two other implementations and report the results. Such inter-operability tests enable the students to detect subtle bugs in their implementations. We also allow students to leverage the results of these tests to update their projects. This improves the quality of their projects and thus their grades. By their nature, these interoperability tests verify that different implementations can interact. If they succeed, it is likely that the students have correctly implemented the protocol. However, these tests do not tell anything about the architecture and the quality of the student's code. In a fourth and final phase of the project, we organize a confidential peer-review phase during which each student evaluates two anonymous implementations from other groups of students. This peer-review is organized through a local instance of the `HotCRP` conference reviewing website. Each student has to answer a set of questions on the quality of the code, its structure and correctness. We intentionally do not ask the students to grade the project from other students in contrast with other colleagues who use peer-reviews for grading. Once all of these phases are over, the teaching staff thus receives for each project: *i)* a complete implementation of the transport protocol; *ii)* an inter-operability report; and *iii)* peer-reviews on the code quality. In addition, a final test report is generated by a series of automated tests defined by the teaching staff upon project submission. Combined, these 4 elements enable the teaching staff to both quickly grade each project and also provide a customized detailed feedback to the students.

Lessons learned. 61% of our students totally agree or agree that these INGInious exercises of the bootstrap phase were sufficient to allow them to start the project.

Many professors would grade the project after the second phase, i.e., right after the implementation phase. Our experience is that by doing so, the students would miss many lessons that experienced network engineers learn by interacting with others. When new protocols are implemented, the engineers who develop the first implementations often organize interoperability events to validate their respective implementations. Such events are a unique learning opportunity for the students as well. The interoperability tests are stressful for the students (71% because it is a strict deadline) but mainly helpful: 74% have discovered errors in their implementation, 72% have identified ideas for improvements by observing the behavior of other implementations and 82% modified their implementation based on these tests.

We use the peer review phase to achieve different pedagogical objectives. First, the peer reviews encourage the students to read code written by other students. Code reviews are an important part of the work of many computer scientists and it is important to train the students to these activities. We insist on constructive comments that can be used by the students who receive them to improve the quality of their project. Second, these peer reviews provide qualitative feedback to the students. Given the size of our classes, we cannot anymore provide personalized feedback to each student and have to rely on automated tests when grading projects. The reviews returned by the students are graded and contribute to a small fraction of the course grade. This phase of peer-review pushes students to be diligent: 65% of students want to get a positive report from their peers. According to students, peer-reviewing is useful. 74% of our students have identified classical errors that they would avoid in the future by reviewing the code of other students. Furthermore, 73% have discovered good practices that they plan to apply to future projects.

We have tested two strategies to allocate the peer reviews to the students. Our first strategy was to randomly allocate two reviews to each student. This strategy is simple to implement and works for most students. Its main advantage is that each project is reviewed by the same number of students. However, it leads to problems with very good or very bad student projects. For such projects, it is very difficult for an average student to provide constructive feedback. We now randomly allocate five projects to each student and then let they select the two of them that they will review. This avoids difficulties with projects that are too weak or too strong, but does not guarantee that all projects receive feedback from other students. As we have no guarantee that all students write the required reviews anyway, this problem is also present with the initial allocation strategy.

3.2 Observing deployed protocols

During the second half of the course, we organize a six weeks long project that encourages the students to improve their understanding of key Internet protocols. This project starts shortly after the presentation of the application layer and progresses in parallel with the course. For this project, each student selects one website and provides some short but precise four-pages two column paper describing how various Internet protocols have been tuned on this website.

Description. Every week, we encourage the students to focus their analysis on one particular aspect of the studied website. At the beggining of each week, we provide guidelines and suggest tools that the students could use. We start from the simpler protocols and then expand to more complex protocols, sometimes requiring special tools. Before this project, the students have been exposed to packet dissectors such as `Wireshark` and the exercises described in Section 2.3. We start usually start by analyzing the configuration of the DNS of studied website. With simple tools such as `dig` or `nslookup`, the students can determine the number of nameservers, the types of DNS records, the support for IPv4 and/or IPv6. Using `traceroute`, they try to determine where the website is hosted. We also provide to the students RIPE Atlas measurements from probes located on different continents to the studied websites. This gives some hints on how those websites rely on Content Distribution Networks. During the recent years, the students have observed the growing importance of IPv6 and DNSSec using these measurements.

The second week is devoted to HTTP. We leverage the developer extensions of the Chrome, Firefox, and Safari browsers and first ask the students to analyze the number of servers that are contacted when browsing a given web page. Students are usually puzzled when they first observe that dozens of websites are usually contacted to render a single web page. They then analyze the HTTP headers to identify the non-standard ones and try to infer their meaning. Another element of the HTTP analysis are the HTTP cookies. The students are usually very surprised by the real utilization of these cookies. During the last years, they have also started to observe the deployment of HTTP/2.

The third week is devoted to the analysis of Transport Layer Security (TLS). Ten years ago, TLS was reserved for banks and small parts of e-commerce websites. During the last years, we have observed an very significant increase in the number of websites that use TLS. During this week, the students use either the developer extensions of their browsers or `openssl`'s `s_client` to analyze how websites negotiate TLS (e.g., version number, proposed cryptographic schemes, and specific parameters). This analysis highlights the complexity of the TLS configuration of large websites and the trade-offs that they have to do. Students have also observed the deployment of TLS 1.3.

During the fourth week, the students observe how TCP is used by large web servers. They analyze the TCP options advertised by the server, verify whether it supports Explicit Congestion Notification, or try to infer its initial congestion window.

After this analysis, the students finalize their report. Our schedule does not enable us to organize a peer-review round for this project, but this would be very valuable. Each student report is reviewed by one professor who provides some short feedback by email and a grade. The best project reports are published on a voluntary basis on our department website and used as examples during the next year.

Lessons learned. Initially, we encouraged each student to select one website to analyze. This offered a strong motivation for the students as they studied a website that they used frequently. However, grading those projects was difficult since all students observed different optimizations. We now ask the students to vote for their most popular websites and select a subset of them so that each website is analyzed by about a dozen of students. This provides a good diversity in what students can observe and simplifies the grading, which is important with classes of 200 students and more.

An important point of this analysis is that students directly observe that the Internet evolves. First, as they have access to a selection of the past student works on other websites[1], they see a trend emerging over the years. Five years ago, the observed websites were using HTTP version 1.1 over IPv4. Today, they often rely on HTTP version 2.0 over TLS and IPv6. Second, as all teaching materials are inevitably subsets and overviews of the protocols and practices used in the Internet, this project allows students to finely observe actual websites. They often find and learn from differences between theory and practices, such as the use of custom HTTP headers and the first deployments of QUIC. 73% of our students confirm that observing protocols on real websites was motivating and pushed them to surpass themselves. Finally, this introduce a feedback loop to the teaching staff that can update the ebook to describe and explain new practices based on what the students observed.

[1]Authors of those works gave explicit consent to their publishing.

3.3 Designing and implementing an enterprise network

This project [17] is part of an advanced networking class. It builds upon the skills learned during the introductory networking class and could be used to conclude such a course.

Description. We start it with a detailed presentation of the campus network by one our network engineers. This 2 hours presentation[2] highlights all the elements that compose a real network (e.g., routing, load-balancers, web servers, monitoring infrastructure, firewalls) and explains many of the design choices they made over the years. Some of these are purely technical, others are based on economic factors. With this background, the students are tasked in groups of four to six to design, build and test an evolution of the campus network in a virtual environment that we provide them. The students have the full freedom to modify the network topology, while still keeping the overall network cost equivalent, and install the software and protocols that they select. We host a copy of the virtual network of each group on a remote server, and setup layer-2 bridges, provide external connectivity advertized through BGP, and a custom DNS TLD. The students can thus build a mini-Internet through BGP peerings (direct or through their connectivity provider that we manage), and build a customized DNS hierarchy. We additionally provide them with a looking glass of what we observe about their network (e.g., advertized prefixes, reachability, latency). At the end of the project, students submit a report on their implementation and design choices, which is then anonymously peer-reviewed by other groups, similarly to the first project (§3.1).

Lessons learned. Leaving them this much freedom forces them to explore various open-source tools to compare their features. This is not so different from the work that network engineers would need to do to procure new switches or new routers. A project typically requires the students to explore new or emerging protocols. We only consider IPv6 and ignore IPv4. Furthermore, we've asked the students to multi-home their campus network to two different providers that use Provider Aggregatable prefixes. This forces the students to explore how these prefixes can be delegated to the end hosts. We encourage them to distribute two IPv6 addresses, one from each provider prefix, to each host. This creates some practical issues with routing and firewall configurations, but enables the students to learn the limits of the current networking technologies. Finally, in order to ensure that students make progress, we schedule weekly appointments of 15 minutes with all groups to provide guidance on eventual blocking points. Once all groups have a running network, we then replace these appointments by physical presence in the lab rooms at well-defined hours.

4 THE BENEFITS OF OPEN-SOURCE

The Internet relies on a wide range of open-source software from operating systems like Linux or FreeBSD to client and server applications. When we started to write the open-source *Computer*

[2]The slides for the 2018 session are available at https://github.com/cnp3/CampusNetwork/blob/master/docs/ucl_network_qhunin.pdf.

Networking: Principles, Protocols and Practice ebook a decade ago, our motivation was to contribute back to the community. Students report that having access to the book sources encourages them to contribute to its improvement (35%) and to look at other open-source projects (37%), which is an unusual way to expose them to the open-source movement.

Students cannot simply learn networking by reading textbooks. They need a variety of activities to master this complex topic. We have developed and released various open education resources which can be used during networking classes. A key element of these resources is that they enable the students to learn from their mistakes, from the simple multiple choice questions, providing instantaneous feedback to the students, to the interoperability tests and the peer-reviews that encourage the students to provide constructive feedback. We encourage other networking educators to fork, adapt and contribute to these open educational resources.

Acknowledgements

We would like to thank all the students, teaching assistants and colleagues who have provided comments and suggestions over the years. Jean-Martin Vlaeminck and Mathieu Xhonneux implemented the socket exercises. Maxime Mawait, Antoine Rime, Augustin Delecluse and Louis Navarre developed several `IPMininet` exercises. This work was partially supported by UCLouvain's OER development fund.

Open Educational Resources

An online copy of the different editions of *Computer Networking: Principles, Protocols and Practice* ebook is freely available at https://www.computer-networking.info. The source code of the book, its exercises, `IPMininet`, as well as software used during the different projects can all be found in the repositories listed at https://github.com/cnp3 under permissive open-source licenses. We use GitHub to receive pull requests against those resources. Any contributors can review them. Contributions are vetted for correctness before being integrated. We encourage educators and students alike to contribute to these resources by translating them in other languages, providing updates to the ebook, or adding new exercises.

REFERENCES

[1] Alberola, J. M., and García-Fornes, A. Using feedback for improving the learning process in programming courses. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje 9*, 2 (2014), 49–56.

[2] Bertsekas, D., and Gallager, R. *Data Networks*. Prentice-Hall Internat. Ed. Prentice-Hall International, 1992.

[3] Bonaventure, O., et al. Computer networking : Principles, protocols and practice, 2019. https://www.computer-networking.info - Accessed Jun-20-2020.

[4] Cardwell, N., Cheng, Y., Brakmo, L., Mathis, M., Raghavan, B., Dukkipati, N., Chu, H.-k. J., Terzis, A., and Herbert, T. packetdrill: Scriptable network stack testing, from sockets to packets. In *2013 USENIX Annual Technical Conference (USENIX ATC 13)* (2013), pp. 213–218.

[5] Crooks, T. J. The impact of classroom evaluation practices on students. *Review of educational research 58*, 4 (1988), 438–481.

[6] Derval, G., Gego, A., Reinbold, P., Frantzen, B., and Van Roy, P. Automatic grading of programming exercises in a MOOC using the INGInious platform. *EMOOCS'15* (2015), 86–91.

[7] Gao, L., and Rexford, J. Stable internet routing without global coordination. *IEEE/ACM Transactions on networking 9*, 6 (2001), 681–692.

[8] Hall, B. *Beej's Guide to Network Programming*. Jorgensen Publishing, 2011.

[9] Holterbach, T., Bü, T., Rellstab, T., and Vanbever, L. An open platform to teach how the internet practically works. *ACM SIGCOMM Computer Communication Review 50*, 2 (2020), 45–52.

[10] Kurose, J., and Ross, K. *Computer Networking: A Top-Down Approach, Global Edition*. Pearson Education Limited, 2017.

[11] Lantz, B., Heller, B., and McKeown, N. A network in a laptop: rapid prototyping for software-defined networks. In *Hotnets 2010* (2010).

[12] Piraux, M. INGInious network trace problem. https://github.com/cnp3/INGInious-problems-network-trace, Accessed Jun-20-2020.

[13] Pizzonia, M., and Rimondini, M. Netkit: network emulation for education. *Software: Practice and Experience 46*, 2 (2016), 133–165.

[14] Singer, L. M., and Alexander, P. A. Reading on paper and digitally: What the past decades of empirical research reveal. *Review of Educational Research 87*, 6 (2017), 1007–1041.

[15] Stevens, R., Fenner, B., and Rudoff, A. *UNIX network programming*, vol. 1. Addison-Wesley Professional, 2004.

[16] Tanenbaum, A., and Wetherall, D. *Computer Networks*. Pearson. Pearson, 2013.

[17] Tilmans, O., and Jadin, M. Campus network project. https://github.com/cnp3/CampusNetwork, Accessed Jun-20-2020.

[18] Tilmans, O., and Jadin, M. IPMininet. https://github.com/cnp3/ipmininet, Accessed Jun-20-2020.