Learning Networking by Reproducing Research Results

Lisa Yan
Stanford University
yanlisa@stanford.edu

Nick McKeown
Stanford University
nickm@stanford.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

In the past five years, the graduate networking course at Stanford has assigned over 200 students the task of reproducing results from over 40 networking papers. We began the project as a means of teaching both engineering rigor and critical thinking, qualities that are necessary for careers in networking research and industry. We have observed that reproducing research can simultaneously be a tool for education and a means for students to contribute to the networking community. Through this editorial we describe our project in reproducing network research and show through anecdotal evidence that this project is important for both the classroom and the networking community at large, and we hope to encourage other institutions to host similar class projects.

CCS Concepts

•Social and professional topics → **Computing education;** **•Networks** → *Network performance evaluation;*

Keywords

Reproducible research, Teaching computer networks

1. INTRODUCTION

At Stanford, like many other universities, we offer two main networking courses for our computer science students: an introductory undergraduate class where students learn how the Internet works, including the basic principles such as packet-switching, layering, routing, congestion control etc. (CS144: "An Introduction to Computer Networks"), and a graduate class where students interested in careers in networking as engineers or researchers read and discuss 20-30 "classic" research papers (CS244: "Advanced Topics in Networking"). Networking classes covering similar topics are prevalent at many universities around the world. Where networking courses seem to differ most between different universities is in the type of programming assignments students are required to do. For example, in most undergraduate classes it is common for students to write programs that start with the sockets layer, and build *upwards* to create applications and libraries on top. At Stanford—and some other universities—students start at the sockets layer and work their way *down*: Our students build transport layers, routers, and NAT devices in the Mininet environment, then put all the pieces together to download web pages from a public website to their own computer through their NAT designed in their router, using their transport protocol. In our experience, students who experience "building their own Internet" gain a thorough knowledge of how the Internet works, how to read and implement RFCs, and how to build network systems.

For a more advanced graduate class in networking, it is less obvious what the most appropriate programming assignments are. Should students build more advanced pieces of the Internet—such as firewalls, load-balancers, and new transport layers? This has the advantage of giving them more experience building network systems, but lacks a research ingenuity component where they can dream up and test their own ideas. And so it is more common in graduate studies for students to do a more creative open-ended project of their own design, perhaps using a simulator, testbed or analytical tools. In our earlier experience with CS244, we opted for the second style, and had students create open-ended projects of their own design. But we kept finding the projects to be lacking—mostly because it is hard to build a meaningful networking system or a persuasive prototype in such a short time. Often, students picked projects that turned out to be too ambitious, and on an incomplete prototype it was hard to collect meaningful experimental results. As a result, the projects tended to be incremental, and the educational experience of the students seemed to be too susceptible to their choice of project. After all, it is hard enough to build a realistic, interesting, and functioning networking system in a matter of weeks; it is harder still to devise a novel one from scratch and then get it to work.

And so instead, for the past five years, we have experimented with a completely different style of project. Since 2012, students taking CS244 work in pairs on a three week project in which they attempt to reproduce experimental results from published research in prominent networking conferences like SIGCOMM and NSDI. For example, students might reproduce the main experimental results in the Hedera [4], DCTCP [5], or Jellyfish [27] papers. Over the past five years, 200 students have attempted to reproduce published results from 40 papers and reported their findings on a public course blog, *Reproducing Network Research*. Each blog entry details how to rerun and reproduce the student results, in the spirit of encouraging more widespread reproducibility of networking results throughout our community.

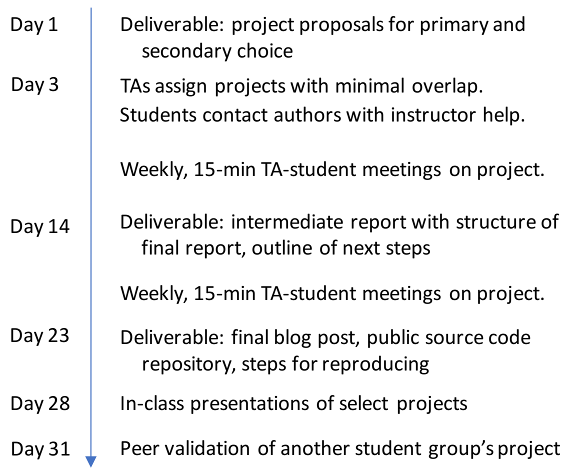
The purpose of this short editorial is to report on our experiences with this style of "reproducing research results" project in a graduate networking class. Specifically, we explain our original goals for this style of project and the educational benefits we hoped for. We describe the wide variety of papers whose results our students tried to reproduce, along with a study of how well they did. We found that a large majority of students were able to successfully recreate the experiment and generate comparable results, with a small fraction unable to. In some cases, they ran into technical difficulties, while in others they were able to make a strong case that the original research contained errors. In all cases, we encouraged students to contact and work with the original authors, which turns out to be a major component of the educational experience for the students. Finally, we report on the educational impact of this project based on interviews from students relating their experiences. We present these findings so that you, the reader, can determine whether this type of project might be useful in your graduate networking classes too.

2. WHY WE CHOSE REPRODUCIBILITY

Our primary over-arching reason for asking graduate students to reproduce published research results is the educational value it brings. Our approach is very similar to how high-school and college students study science worldwide: in tandem with attending lectures and reading textbooks, they reinforce their learning by repeating well-known experiments in the lab. Although the students know and anticipate the experimental outcomes prior to entering the lab, it is widely agreed that the process of reproducing experiments gives students a much deeper understanding of the underlying concepts. Our main goal for adapting this scientific approach to our networking class is for students to obtain a detailed, in-depth understanding of a significant paper, its key ideas, and its key results.

The second biggest benefit is the experience our students get building—or *recreating*—the experiment for themselves. In the science community, reproducing research generally means repeating the experiment and reproducing results identical to the original. In our class, however, students spend much more time building and recreating the original experiment than they do collecting and verifying the results. In our experience, recreating the experiments is the most time-intensive and most fulfilling aspect of the project for our students; achieving identical results is something they may (or may not) do at the end, after their experiment is working. We therefore distinguish the initial step of *recreating* the experimental infrastructure from the second step of collecting and possibly *reproducing* the same results as the original authors. We rate students highly if they successfully recreate the experiment, regardless of whether they can reproduce the same results. In fact, we find that students learn a huge amount when their experiments yield different results from the original research: they must figure out where the discrepancies lie and discern if there are unstated assumptions or inaccuracies in their own results or the published results. This is a fascinating and educational experience, and often a good lesson in diplomacy.

There are many additional benefits to repeating experiments: if students spend a lot of time studying and repeating a published experiment, it leads them to ask "meta" questions about the paper: Why did the researchers pose the problem they did? Why did they use or build a particular prototype or simulator, and why did they collect a specific set of results? These questions allow students to get into the heads of what the researchers were thinking about when they did the research, much more than by simply reading the paper. By going through the process of reproducing results, students gain a deeper understanding of the research process.


Day 1 Deliverable: project proposals for primary and secondary choice

Day 3 TAs assign projects with minimal overlap. Students contact authors with instructor help.

Weekly, 15-min TA-student meetings on project.

Day 14 Deliverable: intermediate report with structure of final report, outline of next steps

Weekly, 15-min TA-student meetings on project.

Day 23 Deliverable: final blog post, public source code repository, steps for reproducing

Day 28 In-class presentations of select projects

Day 31 Peer validation of another student group's project

Figure 1: Student project timeline.

The project also gives students the necessary experience of building a novel prototype, system, emulator or simulator, without necessarily having to be the first one to come up with the idea and try it out. At some level, they already know the idea was a good one: it is practical and has some value, at least enough to warrant publication at a top conference. They are not taking on as big a risk as they would when coming up with their own research problem. As a result, we can expect far more students to obtain satisfactory results. With a high degree of confidence, they already know interesting results are possible, which encourages them (or perhaps goads them via peer pressure) to complete the work.

We also believe it instills an important principle in our future researchers that their research results should be reproducible by others, whenever possible. If results can be reproduced then it is more likely that industry will adopt them, or that other researchers will build upon them - perhaps by directly reusing the experiment's software. There is a growing movement in systems research to make our results more easily reproduced by others [7, 8, 16]. Our students add to the corpus of reproduced results by providing a simple, packaged reproduction experiment; in this manner, they can encourage the whole community to make results more reproducible by others.

All of these reasons seem valuable to graduate students preparing for a career in networking systems research or in industry.

3. THE REPRODUCIBILITY PROJECT

Our students work in pairs and have three weeks (out of a ten-week course) to complete the assignment. They then have an additional week to verify each other's projects and give in-class presentations. Figure 1 shows the project timeline; we describe the main steps of the project below.

Select a project. Each student pair starts by choosing a figure or table from a research paper of interest that is integral to the paper's motivation or claims. This may include comparing the performance of an algorithm against existing algorithms, demonstrating a metric's usefulness, or recording important traffic and workload data. To get the students started, we provide a list of suggested conferences and research publications that we think make good examples, and we encourage students to choose more recent works, or ones that have not yet been attempted by students in previous course offerings. At Stanford we have had students successfully reproduce results ranging from widely cited papers such as Hedera [4] and DCTCP [5] to traditional papers like RED [13], to cutting-edge, as-yet unpublished work like SPDY [1].

Choose a method of reproduction. We encourage students to use either the Mininet [22] or Mahimahi [23] emulation systems for their experiment platform, largely because they are most familiar to the instructors. Mininet is best suited for multi-node topologies, whereas Mahimahi is good when modifying and testing congestion control protocols running over a single link. While we generally prefer students to use emulators—as emulators exhibit more realistic network characteristics, such as real-time, live traffic handling for a given node topology [16]—we also encourage the use of simulators, such as ns-3 [3], if the scale or performance is beyond the reach of an emulator. We provide all students with computing resources on Amazon Web Service (AWS) Elastic Compute Cloud (EC2) to run their experiments, making it easier for others to replicate.

Contact original authors. After deciding which experiments to run, we help the students contact the authors. Opening up this communication channel between students and researchers has two main benefits: the first is for the student, who now has a primary source to contact regarding the tools, setup, workload and use-cases of the given experiment or research tool. The second is for the researcher, who is now aware that his or her work is being analyzed critically; upon completion of the students' experiments, the researcher will have additional feedback on the benefits, caveats, and persistence of his or her findings. We discuss anecdotal evidence on the importance of this communication later in Section 5.

Work with instructors and peers. Recreating other researchers' work is non-trivial; it is essential that course staff support the students throughout their task. In our course of 40 students we were fortunate to have two teaching assistants, who met every group every week, to check-in and provide guidance. In some cases, we were able to pair up students with graduate student mentors whose expertise overlapped with the target research project. We also require a short intermediate report in the middle of the assignment where students describe what they have done so far, and what they plan to do for the remaining time. This allows instructors to give feedback to the students on the feasibility of any remaining steps.

The course ends with students giving short talks about their projects. Students present the main highlights of their reproduced research to the whole class for ten minutes, followed by a short Q&A session.

Write a public blog. Each group is required to document their project—successful or unsuccessful—and any additional findings or conclusions in a public blog post on the course's *Reproducing Network Research* blog. The blog entry must contain all the code and workload in order for someone else to easily repeat the experiments too. And many do; over the years, our website has been visited by the authors of the original research paper, reviewing our students' work, and by new researchers looking for ideas or ways to get started in their own research (see Section 5 for anecdotes).

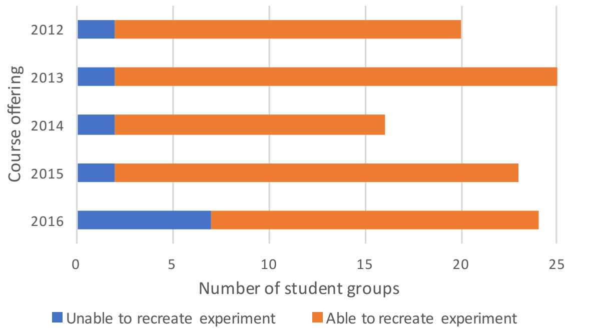


Figure 2: The number of successful student projects, listed by course year. Success is defined as being able to recreate the experiment and generate comparable results.

We verify the results in every blog post using peer validation: every student group is required to replicate the results of another student group. The reproduction effort is required to be an easy, two-step process: (1) download and install any code, and (2) click "run." All code must be available in public code repositories. The students therefore provide all their software source code, experimental data, the means to generate the results, and a detailed interpretation of their results to other researchers. They also upload a public snapshot of their Amazon EC2 machine for easy installation and setup. The public code repositories have proven beneficial for other researchers, who contact the students through the blog in order to use these selected research projects as a base of inspiration or comparison for their own work. These requirements ensure others can build on our students' results, furthering our goal to make more network systems research reproducible.

4. OVERVIEW OF REPRODUCTION RESULTS

Since 2012, we have seen over a hundred student projects in reproducing networking research. Most have been successful—and a few have not—but overall we have observed that students walk away with the confidence that they can overcome difficult, technical challenges in networking research. In this section, we summarize our experiences in more detail.

Figure 2 reports how many student projects successfully recreated research experiments each year, where success is defined as being able to recreate the experiment and generate a result comparable to the original research. The graph shows that a small number of projects each year consistently fall into the "unsuccessful" category, often because students can be over-ambitious: they attempt reproductions in emulators unsuitable for the project, they cannot find the right tools in time, or they overestimate their abilities to build a system from scratch. There are a few other reasons that we discuss later (Section 4.1).

The most popular research papers selected by students are shown in Figure 3. These papers were most likely selected because of their ease of setup in the emulators we chose;

Publication	Reproducibility	
	Able	Unable
TCP Opt-ack Attack [26]	6	2
Jellyfish [27]	8	-
Init CWND [12]	7	-
TCP Fast Open [24]	7	-
Low-rate TCP DoS [21]	4	-
MPTCP [25]	6	-
RCP [11]	4	1
DCTCP [5]	5	-
HTTP-based Video Streaming [18]	5	-
DCell [15]	3	1
Hedera [4]	4	-
Mosh [28]	4	-
PCC [10]	4	-
pFabric [6]	2	1
Sprout [29]	3	-

Figure 3: The 15 most popular research papers selected for student projects.

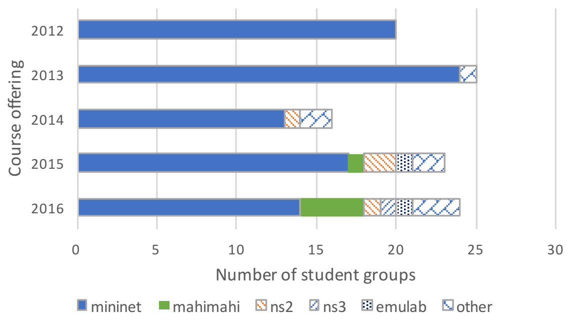


Figure 4: Emulator and simulator platforms used by students for reproducing research, listed by course year.

most of them are variations of TCP, and some of them are application-based or topology-based. Some experiments are more difficult to recreate than others, even if they are from the same research paper; this accounts for some of the unsuccessful projects in Figure 3. Other students are more ambitious in their project, opting to port an existing experiment to a different emulator, which often leads to more difficulties.

Figure 4 summarizes the variety of emulators and simulators that students have used. While we encouraged the use of Mininet [22] and Mahimahi [23], some groups used ns-2 [19], ns-3 [3], and Emulab [17] instead, usually because the original research used these platforms too, making it easier for the students to re-use existing open source code. In some cases, students who started out using simulators ported their experiments to an emulator in order to get real-time, realistic results, all within the three week time span of the project.

Availability of research code. Running an experiment typically requires two components: the system and the experiment workload. Students often obtain both from the authors, or they find them in online, open-source repositories; sometimes, they need to implement it themselves based on the in-depth description in a paper or technical report. Overall, we have found the availability of the original experimental code and workload plays a large part in determining the likely success of reproducing results.

System source code		Workload generation	
Open-source	12	Open-source	9
Open-source but out-of-date	6	Sufficient details in paper	17
Open-source but inconsistent w/results	1	Student-created	14
Contacted author	2		
Binary available	1		
Student-created	9		
Not needed	9		

Figure 5: Availability of source code and workload generation code for each paper.

A summary of the availability of code and data for each of the forty unique research papers studied by students in our course is shown in Figure 5. Occasionally, the research paper lacked key numbers or details about the experiment environment, so students had to reason about additional features and generate their own network workloads. Sometimes, the system source code was open-sourced, but upon further inspection the students found the results of the open-sourced code inconsistent with those published in the paper, and they had to resort to developing the system from scratch. Despite these setbacks, we have found that students who designed their own experiments gained expert intuition in how their system operated and were thus often very successful in recreating experiments.

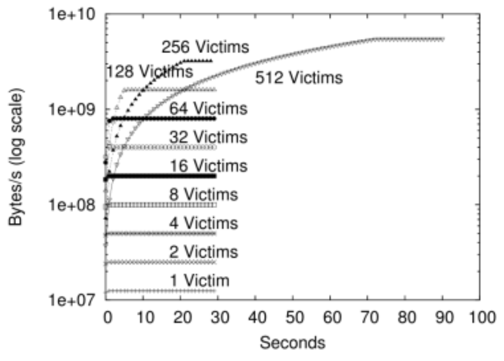
If they are running experiments in an emulator, students typically need to scale the experiment (size or data-rate) so the emulate can keep up. For example, some research results are gathered in large datacenters with hundreds of nodes and link speed of 10-100Gb/s. A typical emulator can handle up to tens or hundreds of nodes, with links running at 1-10Gb/s at most.

4.1 Project successes

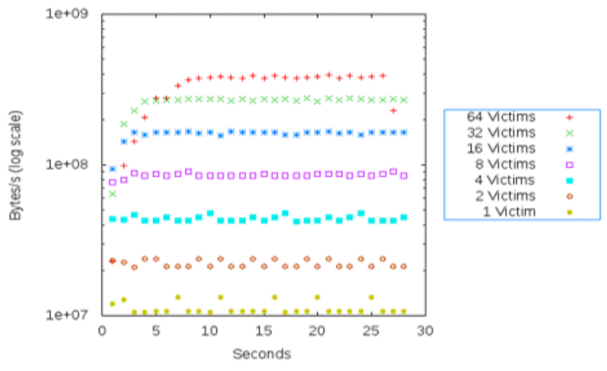
Students have varying levels of success with recreating research results. Due to the complexity of the project, it is an accomplishment in itself for students to simply get the system up and running. We therefore have defined success in this project based on three criteria:

1. Are the students able to recreate the experiment?
2. Are the student-generated results and the original results similar in shape?
3. Can the students justify any discrepancies in results?

Sometimes, students are able to recreate the original work almost perfectly, subject to scaling or computation resource limits. One student group replicated a TCP opt-ack attack [26], where the task was to create a TCP attacker sending optimistic acknowledgements (opt-acks) to multiple victims over a bottleneck link, generating enough traffic to cause congestion collapse (Figure 6a). Even though the original experiment was simulated in ns-2, the students decided to emulate the experiment in Mininet by first designing a Mininet topology and then programming their own opt-ack attacker in Python. They also had to adjust IP table and ARP cache settings on Linux in order to send raw socket traffic on an Amazon EC2 instance. Finally, they were able to produce Figure 6b, which shows very similar traffic patterns to the original, simulated experiment. They explained discrepancies in their results; in particular, they were unable



(a) .



(b)

Figure 6: A successfully recreated experiment: (a) author results (Figure 7 in the original paper [26]) and (b) student-recreated results for maximum traffic induced by a TCP opt-ack attacker over time for multiple connected victims.

to recreate the attack for more than 64 victims due to performance limitations on an emulator for even the largest Amazon EC2 instance with the highest compute power. They also noted that their emulated results had a more jagged shape than the original results, perhaps due to artifacts in measurement. Overall, because the students were able to generate emulated results very similar to the original paper's simulations—and gave sufficient justification for any differences—we consider the student project a success.

Occasionally, students identify discrepancies with the original results for other reasons, despite high confidence in their own recreation of the experiment. For example, one student group compared the performance of ECMP and Hedera [4] on both a hardware testbed and on Mininet. After contacting the original authors, the students reran the benchmark tests and were able to exactly recreate the performance characteristics of Hedera in both the hardware and emulated environments. However, the students consistently found their own hardware ECMP performed significantly better than the original paper's ECMP results. The students reran the ECMP results with spanning tree enabled (something you would not expect in a data-center) and discovered that the resulting, worse performance was identical to the results in the paper. They subsequently contacted the authors to see if they could verify their findings, but the original testbed had been torn down years ago, and there was no way to rerun the experiment for additional verification [16]. As one of the students reflected, "when you create a new testbed

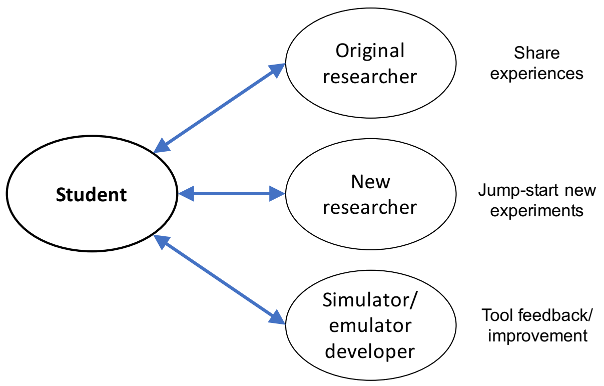

Original researcher
Share experiences
Student
New researcher
Jump-start new experiments
Simulator/ emulator developer
Tool feedback/ improvement

Figure 7: Influences of student project on other parts of networking community.

and create a new environment, there is no way to ascertain the truths or possibilities of the results. On the other hand, if the original authors had used an emulator, such as Mininet, maybe they could've packaged it...so that other people [could use that setup for] experiments."

Reproducing research (un)successfully. There are also cases where students are not able to achieve all three criteria of success. Sometimes, there are limitations in the emulation environment: while setting up an experiment for QJump [14], a student pair had to engineer multiple queueing disciplines in Mininet, a feature that did not come out-of-box with the emulator. Another group reported issues configuring POX [2] and Mininet in tandem when trying to recreate the switch controller topology in DCell [15]. Other times, the age of a paper can affect modern reproductions. A group attempted to replicate the observation that RED maintains significantly higher throughput than Drop Tail queueing at low queue sizes [13]. However, they found that in most cases, Drop Tail and RED performed equally well. After discussion with a commenter it seems that the most likely reason is the underlying TCP mechanism, which in modern times has evolved considerably, perhaps reducing the relative benefits of these two queueing mechanisms.

Students are also occasionally too ambitious: A pair of students tried to implement the rate-based adaptive video streaming of FastMPC [30] in a popular open-source media player. They began the project well by finding the same video and wireless traces used in the original experiment. However, they ran out of time trying to find an appropriate optimizer that could solve the mixed linear programming model for FastMPC. In retrospect, situations like these could have be avoided with timely interventions by teaching staff, who can help students find appropriate tools, or scale down the scope of their project.

4.2 Participating in the community

An unexpected outcome of this project is an increased role of students in the networking research community. While designing and running the experiments, students had to interact with the original authors, new researchers who came across our course blog, and even developers of the emulators or simulators. We believe the benefits of these interactions go both ways; the networking community at large can also benefit from these student research reproduction projects. We summarize the interactions in Figure 7. Original researchers can share their experiences with students to aid in the reproduction effort, and students can give feedback on how well the system works in different environments. New researchers can use the blog post and public repositories published by students to jump-start new experiments in new research.

Interacting with platform developers. Simulator and emulator developers can also use student projects by treating them as use cases for evaluating their platform utility. If the platform is still in development, these student projects give developers more opportunities to improve their tool. In our course, we recommend Mininet [22] and Mahimahi [23], two emulators that were initially developed at Stanford and MIT, respectively. When we began this project in 2012, we also received a large volume of feedback from students regarding the usability of Mininet-Hifi, an extension of Mininet designed to give more accurate timing information and detect when it fails to faithfully meet timing. Through the process of recreating research, students realize the advantages and disadvantages of the platform they are using to recreate results. They can then critique differences in the results and analyze whether the platform setup influenced the replication. When administering a research reproduction project at other academic institutions, we encourage trying out home-grown tools, as student projects are a valuable way of getting feedback on the robustness and accuracy of these new tools.

5. STUDENT EXPERIENCES

With all of the effort involved in recreating research experiments, what's in it for the students? After looking through course evaluations and blog posts, we invited some students to share how their project experiences shaped their perspectives on networking and research. Overall, the students said that the project allowed them to undertake and understand new networking topics, gain confidence in their own research abilities, and participate in the networking community. We share some anecdotes highlighting each of these experiences below.

Encountering an unknown facet of networking. Networking is a broad area at the intersection of many fields, and often it is difficult for students with domain expertise in computer science to interact with the lower layers of the networking stack without first understanding the principles of electrical engineering and communications. This project is a good way for students to get a quick, in-depth view of unfamiliar networking areas. A third-year undergraduate and his partner were curious about wireless research; having come from computer science backgrounds, neither of them knew what areas of wireless research were often studied, but they selected a recent paper tackling Wifi handovers with MPTCP [9]. As the student reflected, he and his partner chose their particular project because it was the best way to learn about a handover problem they had both experienced as end users. When asked what they felt about the experience, he said, "I liked it. I specifically liked the level of familiarity I got the paper. There's a level you can only get by reproducing it or implementing it." After communicating with the authors, the students were able to run an experiment simulation in ns-2 to confirm results illustrating the throughput of different transport protocols during handover between two Wifi access points. After this initial confidence, they then moved on to extend the author's work to show results for three Wifi access points in a three-dimensional graph.

Understanding cutting-edge research. Senior students are also interested in learning cutting-edge research that will help them generate ideas for their own future projects. Reproducing research on a short timeline is a great way to interact with other researchers and understand how to use common tools without needing to expend the rigorous engineering efforts required to achieve research-level system mastery. A pair of second-year graduate students were inspired to reproduce the results from QJump [14] due to both of their research interests in networked systems. One of the students had attended NSDI 2015 and had heard the authors' presentation in person; at the time, her own research was focused on reducing the latency of networked memory in datacenters, and she felt that QJump was an innovative method for scheduling datacenter traffic. As she recounted, "You could tell from their paper that they really tried to make everything reproducible." The researchers had published methods for recreating experiment workloads for all figures in their NSDI publication.

However, she noted that they ultimately did not use the authors' work directly: "Their assumption was that [people] would reproduce the results in an actual datacenter, whereas we did the emulation in Mininet. In the end, we did not use their scripts directly, but it was nice to see that the authors were enthusiastic to have their work reproduced." This pair of students contacted the authors throughout the project to reconcile scaling and timing differences that arose from using an emulated environment in place of a datacenter and were finally successful in recreating the experiments in Mininet. The other student commented that the original authors even "tweeted about [our final blog post], actually." The overall reproduction effort helped the students understand on a deep level what types of traffic control schemes work in datacenters. The first student mentioned that after the course, she implemented a scheduler for her research similar to one from the project, "which is something that I wouldn't have done if [I had just read] the paper."

Digging deep into workshop papers. Reproducing research also boosts confidence. One first-year graduate student commented on the project selection process, saying, "you don't want to reproduce something that requires a lot of previous knowledge or that is hard to reproduce, and you want something that's interesting to you. That process itself takes some learning." Initially, her group had selected a very complex and ambitious project that would require significant time to engineer; eventually, they selected a workshop paper on a self-clocked rate adaption for multimedia (SCReAM) [20]. Using workshop papers for a student project are sometimes more challenging than using full-length words, perhaps because the former has a shorter, briefer publication format. However, the students were able to use the authors' public repository code as reference and transformed the authors' simulation into a real-time UDP-based solution on Mahimahi. While recreating the experiment, they realized that there were parameters that functioned well for the original simulation but not for their emulation. After adjusting these parameters, the students were finally able to observe the same results in Mahimahi. One of the students reflected that it was surprising to see that "papers written at this level could also be understood by students who have taken only two courses in networking, and results can be reproduced in part."

With this realization, she and her partner were happy with their published course blog post, which they considered an important facet of their contribution to the overall research work: "[From an educational standpoint,] blog posts are easier to read than papers. If there is one cool idea from a paper that you can reproduce and put into a blog post, I think that could be very valuable. Because in a sense you already did the set up for them, and you wrote it in a [more lucid] way." As if confirming her newfound perspective, the authors–whom the students did not contact during their project–incidentally came across our course website and contacted her and her partner, addressing critiques and questions that were raised in the students' project blog post. The students also received emails from another graduate student to ask for additional details on running the experiment for his own research.

Boosting experience for future careers. The process of recreating experiments from research can be useful for fostering career skills for both academia and industry. A now-graduated student who recreated experiments for the congestion control protocol DCTCP [5] mentioned that interacting with Mininet was invaluable in her current industry job in network emulators. She and her partner set about recreating an ns-2 simulation of DCTCP's performance in Mininet and came across setbacks in "the kernel version, the amount of memory, the software version... things that [we] didn't really anticipate having trouble with." However, overcoming these struggles were valuable for her current engineering career; she said that "reading up about Mininet and being familiar with how to use it helped me ramp up [faster in my job] because it happens that my team builds something similar to Mininet." Furthermore, thinking critically about networking papers was a skill that aided her technical conversations with coworkers: the course was "very different from any other course I've taken... [where] you're taught principles, learn how to apply them, and write an exam. This [course, on the other hand] would actually prepare me for the real world.

6. CONCLUSION

In this short editorial, we have highlighted some of our experiences offering a graduate-level networking project in recreating experiments from network research. We have provided a step-by-step guide for an example project in an advanced networking class. We have found that the experience is rewarding and interesting for the students, and it gives them a chance to interact with researchers. In addition, we have learned that documenting the results of these reproduction studies is an essential resource for both future students and the research community at large. We hope that the materials presented in this editorial inspire you to consider offering similar projects in your graduate networking courses too.

7. REFERENCES

[1] Making the web speedier and safer with SPDY. http://googlecode.blogspot.com/2012/01/making-web-speedier-and-safer-with-spdy.html. Accessed: 2017-02-13.

[2] noxrepo/pox: The POX controller. https://github.com/noxrepo/pox. Accessed: 2017-02-22.

[3] ns-3. https://www.nsnam.org/. Accessed: 2017-02-22.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI 10)*, NSDI, pages 19–19, Berkeley, CA, USA, 2010. USENIX Association.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM, pages 63–74, New York, NY, USA, 2010. ACM.

[6] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pfabric: Minimal near-optimal datacenter transport. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM, pages 435–446, New York, NY, USA, 2013. ACM.

[7] C. Boettiger. An introduction to Docker for reproducible research, with examples from the R environment. *CoRR*, abs/1410.0846, 2014.

[8] O. Bonaventure. The January 2017 issue. *SIGCOMM Computer Communication Review*, 47(1):1–3, 2017.

[9] A. Croitoru, D. Niculescu, and C. Raiciu. Towards Wifi mobility without fast handover. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 219–234, Oakland, CA, 2015. USENIX Association.

[10] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira. PCC: Re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 395–408, Oakland, CA, 2015. USENIX Association.

[11] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Compute Communication Review*, 36(1):59–62, Jan. 2006.

[12] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. An argument for increasing TCP's initial congestion window. *SIGCOMM Computer Communication Review*, 40(3):26–33, June 2010.

[13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, Aug. 1993.

[14] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, and J. Crowcroft. Queues don't matter when you can JUMP them! In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 1–14, Oakland, CA, 2015. USENIX Association.

[15] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*,

SIGCOMM, pages 75–86, New York, NY, USA, 2008. ACM.

[16] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT, pages 253–264, New York, NY, USA, 2012. ACM.

[17] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Large-scale virtualization in the Emulab network testbed. In *USENIX 2008 Annual Technical Conference*, ATC, pages 113–128, Berkeley, CA, USA, 2008. USENIX Association.

[18] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC, pages 225–238, New York, NY, USA, 2012. ACM.

[19] T. Issariyakul and E. Hossain. *Introduction to Network Simulator NS2*. Springer Publishing Company, Incorporated, 2nd edition, 2011.

[20] I. Johansson. Self-clocked rate adaptation for conversational video in LTE. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS, pages 51–56, New York, NY, USA, 2014. ACM.

[21] A. Kuzmanovic and E. W. Knightly. Low-rate tcp-targeted denial of service attacks: The shrew vs. the mice and elephants. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM, pages 75–86, New York, NY, USA, 2003. ACM.

[22] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[23] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate record-and-replay for HTTP. In *USENIX Annual Technical Conference (USENIX ATC)*, pages 417–429, Santa Clara, CA, 2015. USENIX Association.

[24] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan. TCP Fast Open. In *Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies*, CoNEXT, pages 21:1–21:12, New York, NY, USA, 2011. ACM.

[25] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? Designing and implementing a deployable multipath TCP. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 399–412, San Jose, CA, 2012. USENIX.

[26] R. Sherwood, B. Bhattacharjee, and R. Braud. Misbehaving TCP receivers can cause internet-wide congestion collapse. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, CCS, New York, NY, USA, 2005. ACM.

[27] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI 12)*, NSDI, pages 17–17, Berkeley, CA, USA, 2012. USENIX Association.

[28] K. Winstein and H. Balakrishnan. Mosh: An interactive remote shell for mobile clients. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 177–182, Boston, MA, 2012. USENIX.

[29] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 459–471, Lombard, IL, 2013. USENIX.

[30] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 325–338, New York, NY, USA, 2015. ACM.