Experience Report: Using the FABRIC Testbed to teach a Graduate Computer Networking course

Alexander Wolosewicz
Illinois Institute of Technology
Chicago, Illinois, USA

Prajwal Somendyapanahalli Venkateshmurthy
Illinois Institute of Technology
Chicago, Illinois, USA

Nik Sultana
Illinois Institute of Technology
Chicago, Illinois, USA

Abstract

The curriculum for a graduate Computer Networking course in Computer Science typically includes activities that help students gain a variety of practical skills that complement the theoretical knowledge they learn during the course. These skills are developed through exercises that present students with scenarios in which they are to understand or cause specific communication behavior over a network. These exercises are constrained by the computer resources that students use for learning. Ideally those resources can be tuned to increase the fidelity of the network that a student is managing—and ultimately allow each student to fully control their own network.

This paper describes the motivation, process, and challenges of delivering a graduate course in networking using resources on FABRIC—a publicly-funded, international testbed for research in networking. The paper analyzes the experience of teaching three graduate courses on networking, and reflects on using FABRIC to (1) ensure that students have equal access to a high-quality network environment (rather than rely on students' individual laptops or self-managed school equipment), and (2) exploit the research testbed's flexibility to develop a rich range of exercises for students. We discuss our lessons learned and share advice for other instructors.

CCS Concepts

• **Social and professional topics** → **Computing education**; • **Networks** → Programmable networks;

Keywords

Research testbed, Computer networking, Jupyter notebooks

ACM Reference Format:
Alexander Wolosewicz, Prajwal Somendyapanahalli Venkateshmurthy, and Nik Sultana. 2025. Experience Report: Using the FABRIC Testbed to teach a Graduate Computer Networking course. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641554.3701923

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0531-1/25/02
https://doi.org/10.1145/3641554.3701923

1 Introduction

Students who take a graduate Computer Networking course in Computer Science learn about the exciting combination of technologies that support the communication, business, and government platforms that many use daily. The curriculum of such a course has long been composed of teaching both theoretical and practical skills [27].

Practical skills are developed through exercises that present students with increasingly complex scenarios in which they are to understand or cause specific communication behavior over a network. Examples of exercises include: configuring a network formed of a set of devices, measuring the performance of transfers between those devices, analyzing network traffic to determine communication patterns, diagnosing connection problems, and writing software that uses the network.

Students use computer resources to carry out these exercises. Examples of resources include: **(1)** a student's own laptop on which each student virtualizes [24, 28] or simulates [33] a network on which they carry out the exercises; **(2)** a university-provided server on which the virtualization or simulation software is installed; or **(3)** resources provided by a third-party, such as a publicly-funded testbed or a cloud service like AWS or Azure.

In this paper, we describe the motivation, process, and challenges of delivering a graduate course on Computer Networking that uses resources on FABRIC. In Section 4, we analyze the experience of teaching three graduate courses on Computer Networking over three years. Each course made a different choice between **(1-3)**. These choices are analyzed in Section 5. We found that choice **(1)** disadvantages students with older laptops and causes a maintenance burden for instructors to support different laptop platforms (e.g., Apple Silicon vs x86); choice **(2)** does not scale well with class size and exercise complexity; and choice **(3)** involved sharing the network infrastructure with several other users, but this option proved to be the best in our experience. For choice **(3)** we used FABRIC [12]—a publicly-funded, international testbed for research in networking. Section 6 describes the design of the course that uses FABRIC, and Section 7 discusses lessons learned and shares advice for other instructors.

The FABRIC testbed has been under development for several years, but only entered its operational phase in October 2023. FABRIC encourages using the testbed for teaching [4]. It provides a Jupyter notebook interface [25] for testbed users to describe network configurations and computations. This enabled us to teach through a form of literate programming [26]—where code was interspersed with explanation—and use live coding [34] in lectures.

All network and configuration operations can be done by writing Python code in a Jupyter notebook, with which students were

already familiar. This Python interface encapsulates configuration and invocation commands that are executed directly on the network—allowing the skills learned in this course to transfer to non-FABRIC settings. FABRIC handles the allocation of resources, and allows each student to fully control their own network. This minimizes the operation and maintenance burden for instructors, and ensures that students have equal access to a high-quality network environment that can be used through older laptops.

FABRIC scaled well with class size, and its flexibility enabled us to develop a rich range of exercises for students. This included exercises on recent topics in Computer Networking, such as Software-Defined Networking (SDN) [21], P4 programming [17], and BPF [30].

As an unplanned benefit, we found that using a research testbed provided an excellent stepping stone for students who become interested in doing research in computer networking after the course, or in learning more specialized skills. In our case, two students went on to carry out independent study projects by building on the assignments and skills they developed during the course, and used FABRIC as their development and testing environment.

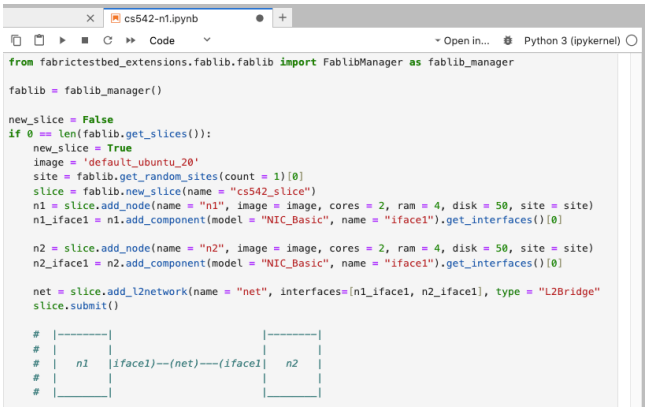

Figure 1: The first Jupyter notebook in the course shows how to set up a *slice* (a set of resources forming a student's network), configure those resources, and use that network. This is done using FABRIC's Python-based API.

2 Background: FABRIC Testbed

FABRIC [12] is a distributed testbed [11] that spans 33 sites—29 in the US, 3 in Europe, and 1 in Asia. FABRIC users can allocate "slices" of resources across all these sites. FABRIC provides high-capacity network resources [7] and it is part of a long-term, international effort to advance our understanding of networks and their application in research [11, 15, 31].

Access to FABRIC is mediated by CILogon [13], which streamlines the authentication process for academic institutions. FABRIC uses a JupyterHub instance to host Jupyter notebooks [25], and it is configured by writing Python programs as shown in Figure 1. FABRIC also offers a web interface. Figure 2 shows the slice that was started in Figure 1 being visualized on this web interface.

Once a slice is started, users connect to individual nodes by using Secure Shell (SSH). Nodes consist of Virtual Machines (VMs) that are started in one of the FABRIC sites, and in which software can be

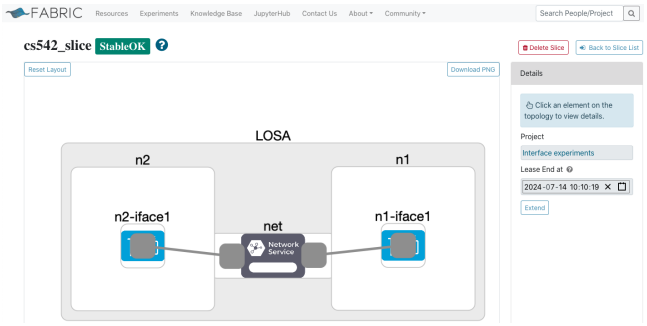

Figure 2: FABRIC visualization of the network topology created using the notebook in Figure 1. Nodes "n1" and "n2" are both linked to the "net" network through their network interfaces ("n1-iface1" and "n2-iface1"). These resources were allocated in the Los Angeles (LOSA) FABRIC site.

installed for research or learning. Different slices are isolated from one another, and different users' resources do not interfere with one another. Nodes are accessed through a *bastion* host—or "jump server"—to securely separate FABRIC resources from the Internet.

3 Related Work

The teaching of computer network is an active research area [35] that is driven by the need to make learning material accessible, engaging, and comprehensive. Teaching on testbeds is not a new idea [39]. The contribution of this paper consists of (1) an analysis of the features provided by FABRIC for teaching (Section 5), the design of a course that heavily integrates with FABRIC (Section 6), and a discussion of opportunities, challenges, and recommendations for other instructors (Section 7).

Zhuang et al. [40] describe a cloud computing course that uses a combination of distributed testbeds: PlanetLab [19], Emulab [29], Seattle [18], and GENICloud [14]. In comparison, this paper analyzes the use of FABRIC to teach a graduate networking course. PlanetLab and GENICloud are no longer operated, Emulab provides more frugal resources compared to FABRIC. Seattle relies on contributed resources that have different capabilities—in comparison, FABRIC has a dedicated pool of similarly-resourced hardware, which supports a more consistent usage experience.

SEED [20] emulates Internet infrastructure—going beyond the emulation of the network to include services that manage the network, such as BGP. It presents a Python interface and visualizations. SEED is complementary to this paper's approach, which leverages FABRIC's abundant access to lower-level network resources. Bonaventure et al. [16] describe an online, interactive textbook for networking that extends Mininet [28] running in a single VM. Bonaventure et al.'s work is complementary to this paper, and in the future it would be interesting to port exercises from SEED and Bonaventure et al.'s textbook to run on FABRIC.

Pan [32] and Gomez et al. [23] describe teaching infrastructure that uses private university resources. In comparison, this paper focuses on the use of publicly-funded, shared infrastructure.

4 Platform choices across three courses

This section outlines the exercise platforms that were used for three graduate courses on Computer Networking. The courses differed by class size and course type: the first two courses were project- and reading-based seminar courses with no exams, while the third course was a regular course with assignments and exams.

Over the period in which these courses were taught, we gravitated towards the use of FABRIC because of the advantages it provides both students and instructors—this will be analyzed in Section 5. Each course was taught once during a three-year period, and the experience of each course informed the logistics for the following course. That is, **(C1)** was taught during the first year, **(C2)** during the second year, and **(C3)** during the third year. Table 1 summarizes this information.

Course C1 [6]. This was a small graduate seminar course that focused on scalability in networking. Students were assessed through two individual projects. The course had no exams. The projects consisted of programs that controlled the network's behavior. Students could propose their own projects and would give weekly updates and demos to the whole group. During **(C1)**, students developed the projects entirely on their laptops (see Table 1) using Hangar [36], a Virtual Machine (VM) that was provided by the instructor. The VM packaged together all the software that the students would need for the course, to simplify the setup for students. The VM was used to ensure uniformity in the setup of different students—that is, to avoid different behavior on different students' laptops because of different versions of software being used.

Course C2 [8]. This was a larger graduate course that focused on Software-Defined Networking (SDN). 50% of the course assessment consisted of a project, and the rest consisted of assignments and exams. Students worked on an individual project of their choice during the second half of the course. The project involved a substantial design and programming task to control network behavior. The students used an Internet-connected, university-provided server on which to develop their assignments and project (see Table 1). For each student, this server ran an instance of Hangar [36], the VM used in **(C1)**.

Course C3 [10]. While **(C1)** and **(C2)** were electives focusing on advanced topics, **(C3)** was a general, core course. It did not involve a project, and assessment consisted of exams and assignments. The students used FABRIC to develop their assignments (see Table 1). FABRIC was also used as a demo platform during teaching, to walk students through examples. Section 6 describes the design of the course in more detail. Instead of using a VM like in **(C1)** and **(C2)**, students were provided with Jupyter notebooks in **(C3)**.

Course Type	# Students	Platform for exercises
(C1) Seminar	< 10	Individual student laptops
(C2) Seminar	< 15	University server
(C3) Regular	> 50	FABRIC testbed

Table 1: Course types, class sizes (number of students), and exercise platform used in courses (C1), (C2) and (C3).

Quality	(C1)	(C2)	(C3)
Low-Effort (Student)	✔	✔	✔
Low-Effort (Instructor)	✔		✔
Inclusion		✔	✔
Scalability	✔		✔
Usability	✔	✔	✔
Isolation	✔	✔	✔
Flexibility	✔	✔	✔
Low cost	✔		✔
"Grows with the student"	✔		✔

Table 2: Analysis of platform choices—see Section 5. Color is alternated (black and gray) to aid readability.

These notebooks would then instantiate and configure VMs on FABRIC for use by each student. Each student controlled their own resources.

5 Analysis of platform choices

This section analyzes the platform choices described in Section 4 before the rest of the paper zooms in on the opportunities and challenges of the platform used in **(C3)**. To streamline the presentation, we will use "**(C*i*)**" both to reference the *course* **(C*i*)** and the *platform* used in course **(C*i*)**—the exact denotation will be clear from context.

Table 2 summarizes the comparison of qualities of the three platform choices. Each quality will be explained next.

Low-Effort (Student): For a student, a low-effort platform is easy to access and use. **(C1-3)** are all low-effort for students, since they involve standard, mature tools. Accessing and using those tools is well-understood by graduate students in Computer Science, since the students would have encountered those tools in other courses that involve using VM software and using the Operating System shell.

Low-Effort (Instructor): For an instructor, a low-effort platform is easy to provide to students. In **(C1)**, this consists of telling students where to download the VM. In **(C3)**, this consists of adding students' university email addresses to FABRIC. In **(C2)**, this consists of manually adding each student to access a machine that is hosting VMs and instantiating a VM for each student. Occasionally a student might damage their VM (e.g., by deleting important files), and the instructor will need to recreate that student's VM.

Inclusion: This means that all students have the same (excellent) access to the resource, and they can use it from anywhere. **(C2)** and **(C3)** are accessible over the Internet, and all students get equal access to resources. **(C1)** is sensitive to students' laptop resources—students with older laptops are at a disadvantage. Newer laptops also caused challenges: shortly before **(C1)** began, Apple released their M1 processor. We found that the VM software we were using had not yet been ported to M1, and this necessitated the VM to be converted to use a different type of software.

Scalability: **(C1)** scales with class size (since every student has access to a computer on which they can install the VM). The size of the VM we used was around 2GB. Larger VMs could

be more difficult for students to accommodate, since the VMs would require more resources from students' machines. **(C3)** scales well too—FABRIC has more than enough resources to accommodate the class. Scalability of **(C2)** is limited by the server used to host student VMs. We used a server with a 48-core Xeon CPU and 128GB RAM, but with increasing class or VM size, we would need more machines.

Usability: **(C1-3)** all involved using mature interfaces and tools that are familiar to students.

Isolation: This means that one student's use of the resource does not affect or disrupt other students. **(C1)** provides the strongest isolation between students—since each student runs a separate VM on their laptop—but **(C2)** and **(C3)** also provide strong isolation. We monitored the server running **(C2)** and never found it to be close to resource-exhaustion. **(C3)** forces resource-usage to be limited-duration—i.e., slices expire—to ensure that resources are shared more fairly.

Flexibility: **(C1-3)** all provide flexibly-configurable environments, since they all ultimately rely on VMs.

Low cost: For the instructor, **(C1)** and **(C3)** are no cost. **(C2)** involved acquiring, setting up, and configuring a server for the course.

"Grows with the student" **(C1)** and **(C3)** can be used by students after the course has completed. **(C2)** is more restricted: since a relatively frugal resource is being shared by several students, its use is more carefully managed to avoid disruption.

*Rationale for the **(C1)→(C2)→(C3)** transition.* Finally, we discuss the transitions from **(C1)** to **(C2)**, and from **(C2)** to **(C3)**. Table 2 shows that the transition from **(C1)** to **(C2)** was *less* optimal. But that transition had one key motivation: improving *inclusion* in the course (even at higher cost and instructor effort) after observing that some students struggled to run the VM on their laptops during **(C1)**. The transition from **(C2)** to **(C3)** was motivated by *scalability* (while retaining inclusion) since the class size was being quadrupled, and fortunately this transition brought several other advantages.

6 Design of a course that uses FABRIC

This section describes a graduate Computer Networking course that makes heavy use of FABRIC—this is course **(C3)** from Section 4. The next section discusses lessons learned and offers recommendations for others who plan to use FABRIC for teaching or learning.

6.1 Course Goals

As a graduate-level, general course on Computer Networking, this course spans various topics. It is designed to help students learn about the theory and practice of networking. It serves as a foundation for more specialized courses on Computer Networking, and as preparation for internships, jobs, and future research that students might pursue. It maximizes the choices that students can make.

The course was designed for a three credit-hour, 16-week semester, and required around nine hours of student effort weekly.

6.2 Course Materials and Student Resources

For a lot of the lecture content, the course relied heavily on the book by Tanenbaum, Feamster, and Wetherall [38]. This provided students with a well-established and mature learning resource.

The course also developed new, cohesive learning materials for using FABRIC. These consisted of notebooks that were used for in-lecture examples and for assignments. They will be described further below. They complement the resources by Fund [22], the FABRIC's forum [2], and teaching resources [4, 5].

For advice and feedback, students could attend office hours with TAs and with the instructor. These office hours were staggered to run three times each week, to maximize opportunities for students to attend.

6.3 Course Structure

The course follows the "bottom-up" approach to teaching networking [35]. This involves starting at the physical layer and progressively building up to higher layers that abstract lower-layer details. This approach was chosen in order to set the foundations for the practical skills that this course sought to develop, which focused on low-level programmability. Setting these foundations would have been delayed had the course proceeded "top-down".

Table 3 shows the course assignments. The first assignment consisted of a simple task to start familiarizing students with FABRIC. Most assignments involved interpreting and writing programs. An initial demo was given during a lecture or provided as a recording before students were given a related assignment. The course had three exams: an initial exam, mid-term exam, and final exam.

6.4 Using FABRIC for Teaching

We used FABRIC in two ways: (1) to give interactive demos during lectures, (2) as a platform for evaluating student assignments.

Demos were used to explain new tools or idioms by example, particularly when teaching students how to use packet capture and generation tools, and the P4 [17] and BPF [30] languages.

FABRIC was also used as a platform for evaluating student assignments to ensure consistency with the environment in which students developed their assignments—this avoids problems where different environments (that use different versions of a library, for example) would produce slightly different behavior. We created "grading notebooks" which help streamline the grading of student assignments that were prepared on FABRIC.

Using live coding [34] during demos helped show students the process of writing programs in a new paradigm. This was made interactive during the lecture, by soliciting student input on how to progress or how to problem-solve an issue.

6.5 Use of FABRIC for Learning

From conversing with students and observing their use of FABRIC, we saw that students quickly adapted to using Jupyter notebooks, and they were comfortable with using Python. FABRIC was available to both in-person and online students, and it was accessible around the clock. It provided all students with the same experience regardless of what type of laptop a student has, or the laptop's age.

Jupyter notebooks served as simple, self-contained resources that students could study and extend. Students would receive notebooks

Assignment	Weight	Time	Description
1. FABRIC account	4%	2	Students created their account on FABRIC.
2. Packet analysis	10%	2	Students were given a packet trace and a set of questions about it.
3. Packet generation	10%	2	Students were tasked with generating a packet trace that fulfilled a given description.
4. P4 Programming	10%	4	Introduction to programming in the P4 language [17].
5. P4 Programming	10%	2	More advanced P4 programming.
6. BPF Filters	10%	2	Students were asked to interpret and generate BPF filters [30].
7. Extra Credit	6%	1.5	Reflecting on difficulties encountered during assignments and how they were solved.

Table 3: Course assignments. *Time* is the interval (in weeks) between an assignment's release and its deadline.

through the university's Learning Management System, upload them to their FABRIC account, and start using them there.

While observing students, we did notice some students struggling. The next section analyzes challenges that students faced, and makes recommendations for mitigations.

7 Lessons Learned and Recommendations

This section reflects on challenges and opportunities related to using FABRIC in a graduate Computer Networking course, based on our observations while running this course. It also compiles recommendations for other instructors who use FABRIC.

7.1 Challenges

We found a small set of common challenges among a subset of students. This section focuses on those challenges. We regularly use FABRIC for research, and we did not experience any difficulties in adapting our use of FABRIC for teaching. We have on-boarded several research students onto FABRIC, and that experience greatly helped our preparation for teaching with FABRIC. Teaching with FABRIC involves on-boarding dozens of students to use FABRIC within a short time window however, and helping them to become familiar with the system. Only one student in the course had used FABRIC before—the vast majority of students were completely new to it. To help students scale this learning curve, we used demos and factored enough time into assignment deadlines—the assignment durations can be seen in Table 3.

The remainder of this section focuses on challenges that were experienced by some students: **(1)** Some students did not respond to an account verification step that is emailed to new users, and their account creation was therefore not completed. Mitigation: We clarified the instructions to students.

(2) Using FABRIC requires creating and using cryptographic keys for authentication. Some students created the keys but did not update their configuration. Without that update, they were unable to use the newly-created keys. **(2.1)** Some students configured the keys incorrectly—for example, swapping two types of keys that were created. **(2.2)** Keys expire, but the symptom of an expired key is a failed authentication, which can create confusion. **(2.3)** When the configuration is incorrect, slice creation takes a long time to fail, because of retries and accumulated timeouts [3]. This keeps students waiting and the unsuccessful authentication outcome is confusing to them. Mitigation: We provided clearer guidance on how to configure keys and how the system behaves, and referred students to FABRIC's documentation on credentials [1]. In a recent version, FABRIC also verifies the setup to provide users with feedback before they start creating slices.

(3) Some students confused the JupyterHub host and FABRIC nodes—see Section 2. For instance, they would change code in the JupyterHub host without updating the code in the node. Or they would only upload code to the JupyterHub host and not to the FABRIC node. Some students initially ran commands in the JupyterHub host that were intended to be executed in a node. Mitigation: Clearer guidance to distinguish the two types of environment.

(4) FABRIC occasionally fails to allocate a slice successfully—a previously-allocated resource might have not yet been fully returned to the pool for allocation to other users, or a site might be found to lack resources. This can happen if a large number of FABRIC users decide to use the same site simultaneously, creating a demand spike for resources at that site. Mitigation: We treat this as "known behavior" (see Section 7.3) and advise students to delete the slice and try again. If a site has insufficient resources, then students are advised to try a different site.

(5) From time to time, some resources on FABRIC might be down because of maintenance or because of an outage. If users attempt to use those resources, then the slice allocation fails. Mitigation: We advise students to check FABRIC's dashboard and forums for outage notifications, and to try recreating the site on different sites. See Section 7.3 for more on this topic.

(6) Slices expire, and their resources are returned to the pool. **(6.1)** Students might return to a slice after it expired, recreating the slice, but not running all intermediate cells in the notebook. This can result in unexpected (and incorrect) output. **(6.2)** JupyterHub tokens can expire, requiring a reset of the hub to regenerate the tokens. Mitigation: Expiration is a feature rather than a bug; it teaches students to make sure their notebook encapsulates everything they need, and that their environment can be recreated later.

(7) Some students tried creating several slices with the same name. This results in an error, since slice names must be unique. Mitigation: We helped students interpret this error, and ensure that a slice is deleted first if it is to be recreated.

7.2 Benefits and Opportunities

(1) Students appreciated the persistence of their notebooks and data on FABRIC's JupyterHub during the semester. They remarked that this made it easier to find everything they needed for this course. **(2)** The environment is disposable—that is, if the VM or notebook

```
[10]: caper_node.execute("./a.out -i test.pcap -n -t")
Packet 0 (captured 210/210):    52 75 21 FC E9 4C 33 61 1A EB 17 C5 7A F9 D6 2A
Packet 1 (captured 86/86):      2F 1D 96 AF F1 63 27 D2 0F 04 5F 79 41 B8 86 A5
Packet 2 (captured 86/86):      43 87 77 61 55 37 0F 4F 60 E9 B5 83 69 2A A9 F7
Packet 3 (captured 86/86):      43 87 77 61 55 37 0F 4F 60 E9 B5 83 69 2A A9 F7
Packet 4 (captured 86/86):      43 87 77 61 55 37 0F 4F 60 E9 B5 83 69 2A A9 F7
Packet 5 (captured 110/110):    DD 21 AF 6A 6C C4 1D B9 03 C4 BE 27 C8 FA F3 D4
Packet 6 (captured 110/110):    6D 30 40 CE 0A 87 36 DA 0E E3 11 45 B7 85 93 24
Packet 7 (captured 122/122):    B9 96 84 B4 EC 3C 00 29 B2 3F FE DB 0E 0F 3B 50
Packet 8 (captured 122/122):    24 9F DD B7 5D 8C D8 CD B2 84 81 B1 78 5E 66 49
Packet 9 (captured 86/86):      44 D4 E8 F6 62 D8 FE 26 C9 94 8C 77 1F 03 4E F2
Packet count: 10
```

Figure 3: Jupyter notebook shows the output obtained by running external tools. This example shows the execution of a custom tool that was compiled on FABRIC after being modified by students as part of an exercise.

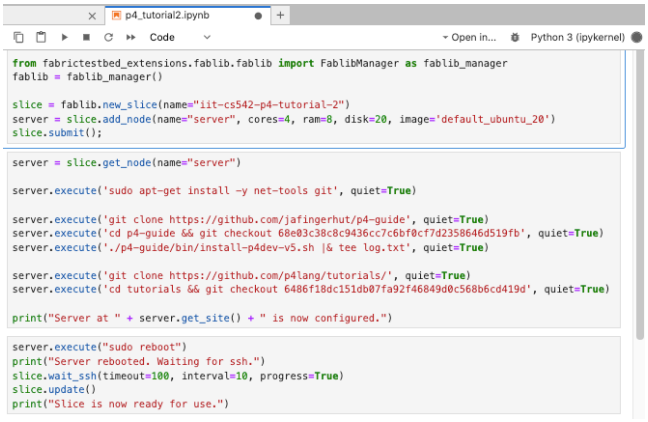

Figure 4: Notebook that sets up P4 development environment.

is damaged by the student, then it is easy to load an earlier copy. **(3)** The repeatability of notebook behavior helped students make progress on their work, and helped them share any problems with us. We would easily reproduce those problems since we were all using the same infrastructure. **(4)** Students found using Jupyter easy and intuitive. Students also remarked about the flexibility with which a network is defined and created on FABRIC. **(5)** Notebooks are segmented into sequences of commands, and students remarked that this helps them make progress in small steps. For example, Figure 3 shows an example where a purpose-built traffic analysis program is executed on FABRIC and returned its output in the notebook. This output shows captured packets and a hexadecimal digest that summarizes the packet. This example also underscores that notebooks could encapsulate everything needed for assignments in this course. **(6)** Students appreciated rapid deployment of complex toolchains. Even when the specific software dependency versions of a demo example or assignment were only obtainable through GitHub (rather than through a packaging system), this could easily be accommodated in a notebook—Figure 4 shows an example of a carefully-tested setup for P4 programming, to ensure that the toolchain would work consistently for students. **(7)** More than one student received support on FABRIC's online forum. Forum answers are discoverable by other students. **(8)** We were struck by students' excitement about learning something new, using a top-of-the-range resource infrastructure, and fresh learning materials.

7.3 Recommendations

The challenges encountered while using FABRIC (Section 7.1) provided great teaching opportunities about the reality of using distributed systems, the best-effort network service model, and learning about industry-standard authentication techniques. It is normal for systems to have maintenance windows, occasional outages, and intermittent faults. The time scales in the course were calibrated to allow plenty of slack for issues to be resolved—particularly for assignment deadlines (see Table 3).

By the same thinking, however, because the system is not always guaranteed to be fully operational, we do not recommend its use for exams at the present time. We considered setting exam tasks that would use FABRIC, and plan to return to this idea when Quality of Service (QoS) constraints are included in FABRIC's slice specifications to guarantee reliability.

In addition to using the JupyterHub interface, FABRIC can make use of locally-hosted Jupyter notebooks running on students' computers. Our research students use both interfaces, but for teaching we found it best to use the JupyterHub interface since (1) it simplifies the setup, and (2) minimizes the reliance on student laptop resources—see "Inclusion" in Section 5.

Breaking down the assignment notebooks in Table 3 into small steps allowed students to make incremental progress. We also recommend providing in-person and recorded demos associated with each assignment, to help students understand the goal of the assignment, and get started.

8 Conclusion and Future Work

Using FABRIC to teach a graduate Computer Networking course provided a great teaching and learning experience, and the challenges we observed could themselves be turned into teaching opportunities. When we teach this course again, we plan to build on this experience and add further course features: **(1)** Cover more advanced material by using the programmable hardware on FABRIC and more diverse experiment examples from scientific networking [37]. **(2)** Use CREASE [9] to provide students with FABRIC-based tools for debugging and diagnosis. Using CREASE's pre-generated VM image on FABRIC also lessens the setup time for students because it avoids students' VMs having to install all the dependencies each time a slice is created. **(3)** Adapt our teaching materials into interactive notebooks in the spirit of Bonaventure et al. [16], but using Jupyter notebooks and FABRIC resources.

Acknowledgments

We thank the students who took our courses, for their feedback. We thank Mert Cevik, Paul Ruth, Michael Stealey, and Komal Thareja from FABRIC/RENCI; Zongming Fei and Jim Griffioen from FABRIC/UKentucky; Jim Basney and Terry Fleury from CILogon/UIUC; and Ilya Baldin from Jefferson Lab for the help they gave us and our students. This work was supported by a URA Visiting Scholars Program Award #24-S-23 and by the National Science Foundation (NSF) under award 2346499. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funders.

References

[1] [n. d.]. FABRIC Credentials Overview. https://learn.fabric-testbed.net/knowledge-base/fabric-credentials-overview/. Last Update: 2023-12-04. Accessed: 2024-07-13.

[2] [n. d.]. FABRIC Educators. https://learn.fabric-testbed.net/forums/forum/fabric-educators/. Accessed: 2024-07-13.

[3] [n. d.]. FABRIC Forum: Slice creation fails. https://learn.fabric-testbed.net/forums/topic/slice-creation-fails/. Last Update: 2024-06-10. Accessed: 2024-07-13.

[4] [n. d.]. FABRIC Information for Instructors. https://learn.fabric-testbed.net/knowledge-base/fabric-information-for-instructors/. Last Update: 2023-08-26. Accessed: 2024-07-13.

[5] [n. d.]. Jupyter Examples for Fabric Testbed. https://github.com/fabric-testbed/jupyter-examples/tree/main. Accessed: 2024-07-13.

[6] 2022. CS595 Designing Large-Scale Networked Systems (Spring 2022). http://www.cs.iit.edu/~nsultana1/teaching/S22CS595/. Accessed: 2024-10-10.

[7] 2022. NSF FABRIC project announces groundbreaking high-speed network infrastructure expansion. https://learn.fabric-testbed.net/knowledge-base/nsf-fabric-project-announces-groundbreaking-high-speed-network-infrastructure-expansion/. Accessed: 2024-01-26.

[8] 2023. CS543 Software-Defined Networking (Spring 2023). http://www.cs.iit.edu/~nsultana1/teaching/S23CS543/. Accessed: 2024-10-10.

[9] 2024. CREASE Project: Causal REasoning and Attestation for Scientific Experimentation. http://crease.cs.iit.edu/. Accessed: 2024-10-10.

[10] 2024. CS542 Computer Networking (Spring 2024). http://www.cs.iit.edu/~nsultana1/teaching/S24CS542/. Accessed: 2024-10-10.

[11] Jay Aikat, Ilya Baldin, Mark Berman, Joe Breen, Richard Brooks, Prasad Calyam, Jeff Chase, Wallace Chase, Russell J. Clark, Chip Elliott, Jim Griffioen, Dijiang Huang, Julio Ibarra, Tom Lehman, Ibrahim Matta, Inder Monga, Christos Papadopoulos, Mike Reiter, Dipankar Raychaudhuri, Glenn Ricart, Robert Ricci, Paul Ruth, Ivan Seskar, Jerry Sobieski, Jacobus Van der Merwe, Kuang-Ching Wang, Tilman Wolf, and Mike Zink. 2018. The Future of Distributed Network Research Infrastructure. *SIGCOMM Comput. Commun. Rev.* 48, 2 (May 2018), 46–51. https://doi.org/10.1145/3213232.3213239

[12] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S. Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. 2019. FABRIC: A National-Scale Programmable Experimental Network Infrastructure. *IEEE Internet Computing* 23, 6 (2019), 38–47. https://doi.org/10.1109/MIC.2019.2958545

[13] Jim Basney, Heather Flanagan, Terry Fleury, Jeff Gaynor, Scott Koranda, and Benn Oshrin. 2019. CILogon: Enabling Federated Identity and Access Management for Scientific Collaborations. In *Proceedings of International Symposium on Grids & Clouds 2019 — PoS(ISGC2019)*, Vol. 351. 031. https://doi.org/10.22323/1.351.0031

[14] Andy Bavier, Yvonne Coady, Tony Mack, Chris Matthews, Joe Mambretti, Rick McGeer, Paul Mueller, Alex Snoeren, and Marco Yuen. 2012. GENICloud and transcloud. In *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit* (San Jose, California, USA) *(FederatedClouds '12)*. Association for Computing Machinery, New York, NY, USA, 13–18. https://doi.org/10.1145/2378975.2378980

[15] Mark Berman, Piet Demeester, Jae Woo Lee, Kiran Nagaraja, Michael Zink, Didier Colle, Dilip Kumar Krishnappa, Dipankar Raychaudhuri, Henning Schulzrinne, Ivan Seskar, and Sachin Sharma. 2015. Future Internets Escape the Simulator. *Commun. ACM* 58, 6 (may 2015), 78–89. https://doi.org/10.1145/2699392

[16] O. Bonaventure, Q. De Coninck, F. Duchêne, A. Gégo, M. Jadin, F. Michel, M. Piraux, C. Poncin, and O. Tilmans. 2020. Open educational resources for computer networking. *SIGCOMM Comput. Commun. Rev.* 50, 3 (jul 2020), 38–45. https://doi.org/10.1145/3411740.3411746

[17] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (jul 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[18] Justin Cappos, Ivan Beschastnikh, Arvind Krishnamurthy, and Tom Anderson. 2009. Seattle: a platform for educational cloud computing. *SIGCSE Bull.* 41, 1 (mar 2009), 111–115. https://doi.org/10.1145/1539024.1508905

[19] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. 2003. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.* 33, 3 (jul 2003), 3–12. https://doi.org/10.1145/956993.956995

[20] Wenliang Du, Honghao Zeng, and Kyungrok Won. 2022. SEED emulator: an internet emulator for research and education. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks* (Austin, Texas) *(HotNets '22)*. Association for Computing Machinery, New York, NY, USA, 101–107. https://doi.org/10.1145/3563766.3564097

[21] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.* 44, 2 (apr 2014), 87–98. https://doi.org/10.1145/2602204.2602219

[22] Fraida Fund. [n. d.]. Teaching on Testbeds. https://teaching-on-testbeds.github.io/. Accessed: 2024-07-13.

[23] Jose Gomez, Elie F. Kfoury, and Jorge Crichigno. 2022. Enabling P4 Hands-on Training in an Academic Cloud. In *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 426–429. https://doi.org/10.1109/DCOSS54816.2022.00077

[24] Rob Jansen, Jim Newsome, and Ryan Wails. 2022. Co-opting Linux Processes for High-Performance Network Simulation. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 327–350. https://www.usenix.org/conference/atc22/presentation/jansen

[25] Jeremiah W. Johnson. 2020. Benefits and Pitfalls of Jupyter Notebooks in the Classroom. In *Proceedings of the 21st Annual Conference on Information Technology Education* (Virtual Event, USA) *(SIGITE '20)*. Association for Computing Machinery, New York, NY, USA, 32–37. https://doi.org/10.1145/3368308.3415397

[26] Donald Ervin Knuth. 1984. Literate Programming. *Comput. J.* 27, 2 (1984), 97–111.

[27] Linda B. Lankewicz. 1998. Resources for teaching computer networks. *SIGCSE Bull.* 30, 1 (mar 1998), 112–116. https://doi.org/10.1145/274790.273173

[28] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (Monterey, California) *(Hotnets-IX)*. Association for Computing Machinery, New York, NY, USA, Article 19, 6 pages. https://doi.org/10.1145/1868447.1868466

[29] W. David Laverell, Zongming Fei, and James N. Griffioen. 2008. Isn't it time you had an emulab? *SIGCSE Bull.* 40, 1 (mar 2008), 246–250. https://doi.org/10.1145/1352322.1352223

[30] Steven McCanne and Van Jacobson. 1993. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX winter*, Vol. 46. 259–270.

[31] Deep Medhi and Peter A. Freeman. 2009. Research challenges in future networks: a report from US-Japan workshop on future networks. *SIGCOMM Comput. Commun. Rev.* 39, 3 (jun 2009), 35–39. https://doi.org/10.1145/1568613.1568621

[32] Jianping Pan. 2010. Teaching computer networks in a real network: the technical perspectives. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 133–137. https://doi.org/10.1145/1734263.1734311

[33] George F. Riley and Thomas R. Henderson. 2010. *The ns-3 Network Simulator*. Springer Berlin Heidelberg, Berlin, Heidelberg, 15–34. https://doi.org/10.1007/978-3-642-12331-3_2

[34] Marc J. Rubin. 2013. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 651–656. https://doi.org/10.1145/2445196.2445388

[35] Jürgen Schönwälder, Timur Friedman, and Aiko Pras. 2017. Using Networks to Teach About Networks (Report on Dagstuhl Seminar #17112). *SIGCOMM Comput. Commun. Rev.* 47, 3 (sep 2017), 40–44. https://doi.org/10.1145/3138808.3138814

[36] Nik Sultana. 2022. Demo: The Hangar environment for Teaching and Research in Programmable Networking. In *30th IEEE International Conference on Network Protocols, ICNP 2022, Lexington, KY, USA, October 30 - Nov. 2, 2022*. IEEE, 1–2. https://doi.org/10.1109/ICNP55882.2022.9940410

[37] Nik Sultana, Yatish Kumar, Chin Guok, James B Kowalkowski, and Michael H L S Wang. 2024. Shape-shifting Elephants: Multi-modal Transport for Integrated Research Infrastructure. In *To appear in the Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets '24)*. Association for Computing Machinery, New York, NY, USA. (Accepted for publication).

[38] A. Tanenbaum, N. Feamster, and D. Wetherall. 2021. *Computer Networks, 6th edition*. Pearson.

[39] Charlie Wiseman, Ken Wong, Tilman Wolf, and Sergey Gorinsky. 2008. Operational experience with a virtual networking laboratory. *SIGCSE Bull.* 40, 1 (mar 2008), 427–431. https://doi.org/10.1145/1352322.1352280

[40] Yanyan Zhuang, Chris Matthews, Stephen Tredger, Steven Ness, Jesse Short-Gershman, Li Ji, Niko Rebenich, Andrew French, Josh Erickson, Kyliah Clarkson, Yvonne Coady, and Rick McGeer. 2014. Taking a walk on the wild side: teaching cloud computing on distributed research testbeds. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 535–540. https://doi.org/10.1145/2538862.2538931