## Public Review for

# An Open Platform to Teach How the Internet Practically Works

Thomas Holterbach, Tobias Bühler, Tino Rellstab, and Laurent Vanbever

This paper describes a software infrastructure that can be used to teach about how the Internet works. The platform aims to be a much smaller, yet representative copy of the Internet. Several of the Internet's key functions are replicated via various pieces of software: OpenvSwitch provides layer 2 functions, FRRouting provides layer 3 routing, and bind9 provides DNS services. These elements reside in Docker containers running Debian Stretch. The infrastructure provides a set of supporting scripts and software tools to simplify creation of topologies out of these components, as well as visualizing their connectivity.

Emulating the entire Internet would be an incredibly challenging undertaking so it should not be surprising this one paper does not completely solve that problem. The abstract and title claim to teach how the "Internet" works but much of the platform's functionality focuses on routing, as opposed to other important aspects of networking such as wireless, DNS, TCP, etc. The paper's description and evaluation is focused on technical aspects of the design, but as a teaching tool it may be more helpful to describe more about pedagogical issues. For example, an evaluation of how much students learn from the tool may be more relevant than a performance evaluation. More details on laboratory assignments, how the approach could be incorporated in classroom activities, a plan for how the approach would be deployed and administered across institutions would have been helpful as well. We also hope the authors can provide a download link and documentation to support others in acquiring and using their platform.

It's easy to look at this paper and find things we wish it had done, but I think that is what is really exciting about it - it makes us dream. The paper is ambitious and visionary, yet extensible - many of the limitations mentioned in the reviews can be addressed in future work by building atop this open platform, conducting further evaluation, etc. Networking educators will benefit from reading this paper as it contains new ideas on how to teach networking in a way that can bring inspiration and excitement to students.

*Public review written by*
**Matthew Caesar**
*UIUC, USA*

# An Open Platform to Teach How the Internet Practically Works

mini-inter.net

Thomas Holterbach
ETH Zurich
thomahol@ethz.ch

Tobias Bühler
ETH Zurich
buehlert@ethz.ch

Tino Rellstab
ETH Zurich
tinor@student.ethz.ch

Laurent Vanbever
ETH Zurich
lvanbever@ethz.ch

## ABSTRACT

Each year at ETH Zurich, around 100 students collectively build and operate their very own Internet infrastructure composed of hundreds of routers and dozens of Autonomous Systems (ASes). Their goal? Enabling Internet-wide connectivity.

We find this class-wide project to be invaluable in teaching our students how the Internet infrastructure *practically* works. Among others, our students have a much deeper understanding of Internet operations alongside their pitfalls. Besides students tend to love the project: clearly the fact that all of them need to cooperate for the entire Internet to work is empowering.

In this paper, we describe the overall design of our teaching platform, how we use it, and interesting lessons we have learnt over the years. We also make our platform openly available [2].

## CCS CONCEPTS

• **Networks** → **Network design principles**; **Network protocols**; *Public Internet*;

## 1 INTRODUCTION

Most undergraduate networking courses, including ours [25], aim at teaching "how the Internet works". For the instructor, this typically means painstakingly going through the TCP/IP protocol stack, one layer at a time, following a bottom-up [19] or top-down approach [13]. At the end of the lecture, students (hopefully) have learnt concepts such as switching, routing, and reliable transport; together with the corresponding protocols.

Learning these concepts is not sufficient to understand how the Internet infrastructure works or, alternatively, why it does *not* work. For this, we think one also needs to understand the ins and outs of how the Internet is operated which includes topics such as network design, network configuration, network monitoring, and… network debugging. Understanding these topics is important as Internet operations tend to have a *huge* impact. Among others, most of the Internet downtimes are due to human-induced errors [18].

We argue that an effective way to teach students about Internet operations—one that we have successfully used for the last four years—is simply to let students operate their own mini-Internet.

***Turning students into operators.*** Each year, for the last four years, around 100 ETH students have built, configured, and monitored an actual Internet infrastructure composed of hundreds of routers split across 60 Autonomous Systems (ASes). Each group of 2–3 students is responsible for administering, from scratch, one AS composed of multiple hosts, layer-2 switches and layer-3 routers. Each network "peers" with others using BGP, either directly or through Internet eXchange Points (IXPs), which we (the instructors) maintain. The students' goal is identical to the ones of actual operators: enabling Internet-wide connectivity, between any pair of IP prefixes, by transiting IP traffic across multiple student networks. As they quickly realize though, achieving this goal is challenging and requires a truly collective effort. We found this to be empowering. The fact that all networks need to work for the Internet as a whole to work really helps to bring together the entire classroom.

Over the years, the mini-Internet project has become a flagship piece of our networking lecture, one that the new students look forward to. Thus far, the feedback we received from the students has been extremely positive, with comments such as: *"It really allows us to apply the theoretical concepts"*; *"I am quite confident about many things on the Internet now"*; and *"It is a unique project"*.

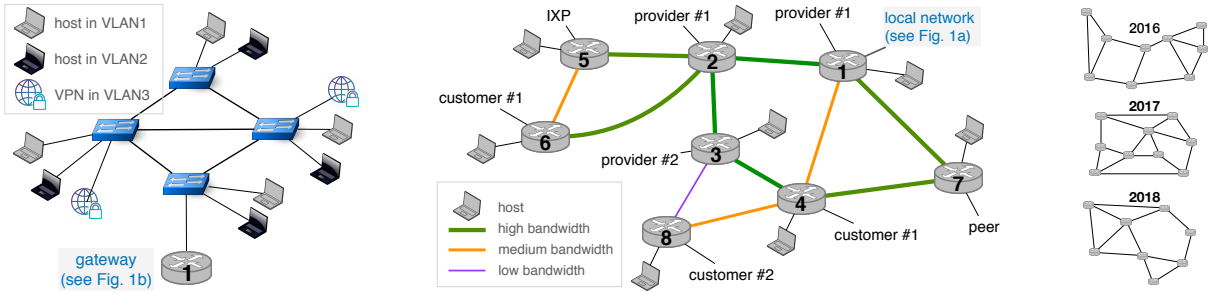
Besides gaining a *much* deeper understanding of the various Internet mechanisms, having students build and maintain their own Internet infrastructure enables them to quickly realize the pitfalls and shortcomings behind Internet operations. Students quickly realize: *(i)* how fragile the Internet infrastructure is and how dependent they are on their neighbors' connectivity; *(ii)* how hard it is to troubleshoot Internet-wide problems; and *(iii)* how difficult it is to coordinate with each other to fix remote problems. Each year, several groups of students come up with proposals (sometimes, even implementations!) to improve Internet operations. These proposals often directly relate to research topics active in our community (such as configuration verification/synthesis or active probing). Perhaps candidly, we believe that encountering operational problems early on in their networking curriculum can help the next-generation of network designers avoid repeating the mistakes made in the past.

***An open platform.*** Given the success of our project, we have open sourced the entire platform [2] and hope that other institutions will start using it. We built our platform with three key goals in mind.

First, we aimed at faithfully emulating the real Internet infrastructure. To do so, we rely on (open-source) switching and routing software implementing the most well-known protocols (e.g., STP, OSPF, BGP). We also rely on virtualization (containers) to interconnect *many* instances (100+) of these software. While relying on virtualization in network education is not new (e.g., [3, 5, 6, 14, 22]), our setting is unique as it is entirely designed to support and facilitate large and collectively-operated routing infrastructures.

Second, while we wanted the students to learn the intricacies of Internet operations, we also wanted to avoid making it too daunting for them. In particular, our students only have four weeks to build the entire mini-Internet. To help them, we developed a suite of troubleshooting tools such as a perfect "looking glass" which allows them to see the routing information of any network, together with a real-time visualization of the overall Internet connectivity.

Third, we wanted the setup to be easy to manage for us (the instructors), flexible (so that we can adapt it each year), cost-effective and scalable (to 100+ students). We therefore automated the entire provisioning: it takes only a few hours to create and launch a




**(a) The L2 network. Hosts are in different VLANs, and router 1 is the gateway.**

**(b) The L3 network. Several routers interconnect hosts and L2 networks. Routers also connect to other ASes or IXPs.**

**(c) Previous L3 topologies.**

**Figure 1: In our mini-Internet project (2019 iteration), each AS has a local L2 network (Fig. 1a) and a L3 network (Fig. 1b). Every year we change the L2 and L3 topology. Fig. 1c shows the L3 topologies we used in the previous years.**

new mini-Internet topology. We also optimized the setup so that it can handle 100+ students on *a single* off-the-shelf server. For larger classrooms, the project can be distributed over multiple servers.

***This paper.*** After providing details on the capabilities of the platform (§2), we show how we use it in our introductory lecture (§3), and describe various pedagogical insights we learnt over the years (§4). We then highlight how we designed the platform (§5), and explain how this design makes the mini-Internet suitable for various teaching objectives (§6) and scalable to large classrooms (§7).

## 2 THE MINI-INTERNET PROJECT

In this section, we first describe the main components (§2.1) before introducing the configuration and monitoring tools (§2.2).

### 2.1 Base components

At the highest level, the mini-Internet is composed of several ASes connected directly or through IXPs (Fig. 2). Each AS is maintained by a distinct group of students and contains several routers, switches, and hosts interconnected through links with configurable bandwidth and delay. Each host can run tools such as `ping`, `traceroute` or `iperf`. When the mini-Internet is correctly configured, any two hosts can communicate with each other.

Within an AS, hosts are connected either to L2 switches or to L3 routers and can be located in different VLANs (Fig. 1a). At least one switch is connected to a L3 router which acts as an IP gateway. As an example, router 1 in Fig. 1b is connected to the local L2 network depicted in Fig. 1a. L3 routers connect to each other internally, but can also connect to routers in other ASes.

In addition, the mini-Internet supports external hosts through the use of L2-VPN servers. Doing so enables the students to connect their own devices to their network.

### 2.2 Configuration and monitoring

Similarly to the real Internet, students configure their network devices through text-based command-line interfaces. To make this task (slightly) less cumbersome, we also provide them with a set of monitoring tools and services.

*Hosts.* All our hosts run Debian Stretch [9] and support traditional commands to measure network connectivity (e.g., `ping` and `traceroute`) and configure the routing tables (e.g., `ip`).

*Switches and routers.* The L2 switches are Open vSwitches [20] while the L3 routers run FRRouting [21]. Both software suites are well-documented, support the main L2 and L3 protocols, and offer similar configuration interfaces than actual switches and routers.

*Looking glass.* In the Internet, operators often rely on "looking glass" services [1] to access the routing tables of remote ASes. Similarly, the students can access a web interface which contains periodically updated routing tables of each router in the mini-Internet.

*Active probing.* Network operators also use measurement platforms (e.g., [17]) to verify the connectivity from an external point towards their AS. In the mini-Internet, students can run `ping` and `traceroute` commands between any two ASes to monitor the connectivity and forwarding paths between them.

*Connectivity Matrix.* Students can access a dynamic webpage which displays whether any two ASes can reach each other as a matrix. The matrix not only gives a good overview of the overall connectivity but also helps pinpointing problems (§3.2).

*DNS.* Finally, we run one DNS server enabling students to use domain names instead of IP addresses.

## 3 THE MINI-INTERNET AT ETH ZURICH

We now explain how we use the mini-Internet in the classroom. We first explain how we design the topology (§3.1), then how we organize the project (§3.2) before describing what we ask the students to do (§3.3).[1] Finally, we explain the limitations we often encounter and how we deal with them (§3.4).

### 3.1 Topology

Our implementation of the mini-Internet allows us to define the topology of the network at every layer, i.e. L2, L3 and AS-level. Fig. 1a shows the L2 topology we used in the lecture in 2019. There are four switches, and each switch is connected to two hosts and possibly a VPN server. One switch is connected to a gateway router. The gateway router belongs to the L3 topology displayed in Fig. 1b which contains eight routers. In addition, one host is connected to each router. Fig. 1c depicts the L3 topologies we used in the previous years. The topology in 2016 resembles the Internet2 topology [12] while the one from 2018 resembles the SWITCH topology [24].

[1] See [2] for the 2019 (and 2020) assignments.

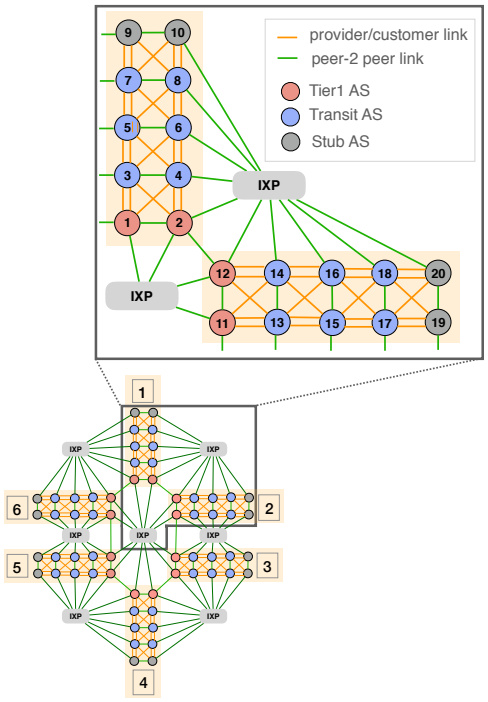



**Figure 2: The topology our students operated in 2019: 60 ASes divided in six regions interconnected via seven IXPs.**

For fairness, each student AS has the same L2 and L3 topology. Fig. 2 depicts how we interconnect these ASes to form the entire mini-Internet in 2019. There are 60 ASes grouped into six different regions. The topology exhibits many of the properties found in the actual Internet: there are Tier1s, stubs and transit ASes, connected through customer/provider and peer-to-peer links. Tier1 ASes are connected in a full-mesh and several IXPs interconnect the different regions in the topology. Every transit AS is connected to exactly two customers, two providers, one peer and one IXP.

## 3.2 Organization

To reduce the student's workload, we group them in teams of three and give each group one transit AS to operate. (We configure the Tier1s and stubs ourselves). We further allocate one /8 prefix to each AS to allocate to their hosts and interfaces.

We divide the project into the three subsequent phases: (i) establishing intra-domain connectivity; (ii) establishing inter-domain connectivity; and (iii) configuring external routing policies. These phases map to different levels of "Internet-wide" connectivity which we depict in Fig. 3 with connectivity matrix (§2.2) snapshots.

First, students have to configure the L2 switches as well as the intra-domain routing so that hosts inside *one* AS can reach each other. As a result, the diagonal cells in the connectivity matrix should turn green. Second, they have to configure iBGP sessions and establish eBGP sessions with their neighboring ASes and IXPs. In the best case, the matrix should now be completely green. Every student group can reach every other group. Finally, we ask the students to configure certain BGP policies e.g., to follow business

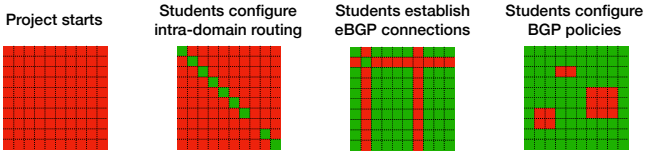



**Figure 3: Snapshots of the connectivity matrix. A green cell indicates connectivity between two ASes.**

relationships. At this point, the matrix often fluctuates as students tend to make mistakes when configuring their policies.

As we can see in Fig. 3, the matrix does not only show the current progress but it also helps us to quickly identify mistakes. For instance, if a cell is red in the diagonal it likely means that the corresponding group has configuration mistakes in their intra-domain routing part. If an entire column is red, the corresponding AS has not properly configured eBGP sessions. Finally, asymmetric patterns often hint towards mistakes in BGP policies.

To guide students through the project, we set up a dedicated online chat room in which students can ask questions, and we organize a Q&A session every week where several TAs provide support. We also organize a "hackathon" in-between the first and the second part where all students meet, discuss with their direct neighbors which IP addresses to use on their external links and figure out what is or is not working. It is also very rewarding for the students to see the matrix turning more and more green as they set up their eBGP sessions. We also leverage the hackathon to perform a live demonstration of the effects of a BGP hijack.

## 3.3 Questions

We now describe some questions we ask our students. Besides answering them, students also need to explain how they verified the correctness of their networks. Note that the students often do not have any prior networking knowledge.

*Configure IP addresses and subnets.* First, students must configure IP addresses and subnets for each host and router interface. To guide them and to simplify the grading process, we ask all the groups to follow the same scheme. For instance, the subnet X.0.200.0/23 should be allocated to the L2 network, where X is their AS number.

*Configure the L2 network.* We then ask the students to configure the local network (see Fig. 1a) to enable direct L2 connectivity between hosts in the same VLAN, but not between hosts in different VLANs. In the latter case the hosts must communicate via the L3 router. Additionally, we ask the students to configure the switch ports in such a way that the final spanning tree follows a certain pattern.

*Configure intra-domain routing.* To enable connectivity within the AS, we ask them to configure OSPF network-wide. At this point, every host in one AS can communicate with any other one in the same AS. We then ask the students to do some traffic engineering. Typically, they should configure load-balancing across different paths. Additionally, we also ask them to implement routing decisions which are not achievable by simple OSPF weight changes.

*Configure inter-domain routing.* This starts during the hackathon. Students must first configure an iBGP full-mesh before setting up the eBGP sessions with their neighboring ASes as well as the IXPs.

*Configure BGP policies.* This is often the most complicated part. We first ask the students to configure the local-preferences as well as the exportation rules to implement the customer/provider and peer-to-peer business relationships with their neighbors [7]. Then, we ask them to implement more advanced policies to influence the inbound or outbound traffic. During this process the students learn about the different BGP attributes (e.g., AS path, MED) and how to use them in order to influence the forwarding behavior. For instance, we often ask them to configure BGP such that the inbound traffic coming from the provider with whom they have two external BGP sessions (see Fig. 1b) arrives preferably via one of the routers.

*Detect and mitigate malicious operations.* Finally, we often ask questions about malicious operations such as BGP hijacking. Typically, we would hijack a part of the IP space of each group and ask the students to identify the malicious AS before mitigating the attack (e.g., by advertising more-specific prefixes).

### 3.4 Limitations

Configuring and monitoring a network is sometimes tricky, and this can become a limitation especially given the limited time budget the students have. First, the students are not familiar with the routers' CLI, and configuring routing protocols is not straightforward for beginners, e.g., we often have questions on how to configure `route-maps`. To help the students we therefore provide additional documentation tailored to the questions we ask.

Second, some students might have persistent misconfigurations, or just start to work on the project late. This affects the connectivity of their neighboring ASes which for instance might not be able to reach some regions in the mini-Internet. Debugging is therefore more difficult and some of our questions cannot be properly answered. To mitigate these problems, we have designed the AS-level topology (see Fig. 2) such that each transit AS always has two providers and two customers to prevent a network-wide loss of connectivity if one neighbor fails. In addition, Tier1, stub ASes and IXPs are automatically configured. This already enables the students to answer most of the questions independently of the other transit ASes. Finally, we also adapt the grading scheme accordingly.

## 4 LESSONS LEARNT

In this section, we describe some of the lessons we learnt over the last four years using the mini-Internet project in our lecture.

***Connectivity is a collective effort.*** It is interesting to observe how the student's perception to connectivity problems changes. At the beginning of the project, they often show us their configuration and ask "what did *we* wrong?". Most of the time their configuration is correct and the problem comes from other ASes. Towards the end of the project though, they tend to first blame other groups before searching the error source in their own solution. As previously explained, we therefore try to build redundant topologies such that the students do not only depend on a single other AS. In addition, we also grade each group on their individual configurations rather than on the overall connectivity. In conclusion, these experiences help the students to understand that an Internet-wide connectivity requires communication between multiple parties.

***BGP is difficult to master.*** Every year, the most confusing topic is the interaction between the control-plane messages (BGP advertisements) and the data plane. Most students have a hard time to realize that they have to e.g., adapt the *outgoing* BGP messages in order to influence the *incoming* traffic. They often wrongly believe route-maps are applied on each IP packet traversing the router. In addition, some students are confused with the language used to configure the routers as it does not follow modern programming language principles. As a result, we improved lecture slides and documentation and put more effort into showing the students the impact of their configuration using e.g., the measurement platform. Overall, we hope these insights help the students in the future to eliminate shortcoming of existing solutions should they have to develop new protocols or ways to configure network devices.

***Automation is key.*** To show the students all the required configuration steps, we do not provide any automation tools. Yet, certain configuration parts are shared between all devices and could therefore be generated automatically. Every year, multiple students submit simple Python or Bash scripts which automatically generate the configuration of all devices in their network. Some students also automate the verification process and e.g., regularly ping each host in the mini-Internet. It is very encouraging to see that the mini-Internet mimics the real Internet infrastructure closely enough such that the students can uncover actual research topics on their own (e.g., configuration synthesis and network verification).

***Visualization is important, but also dangerous.*** Visualization tools such as the connectivity matrix (Fig. 3) are essential for the students (and network operators alike, e.g., [8]) to quickly get an overview of their network connectivity. At the same time though, students often incorrectly assume their configuration is correct as soon as e.g., the matrix lights up in green. As the visualization tools do not reveal *all* the possible problems in the network, relying solely on them is often misleading. To address this problem, we plan on continuing to improve the quality of our visualization solutions in the future releases of the mini-Internet. For example, we plan to implement a web interface that shows the used AS path between two ASes (similarly to [4]) and highlights ASes that do not follow the business relationships.

## 5 IMPLEMENTATION

We implemented our platform in ~3700 lines of Bash and make it publicly available [2]. By default, our implementation runs a mini-Internet with 20 ASes. We provide various configuration files, e.g. to reproduce the L2 and L3 topology depicted in Fig. 1a and Fig. 1b. However the topologies can easily be customized.

In this section, we give more details on how we build the virtual networks, implement the various monitoring and debugging tools and explain how the students can access the mini-Internet.

***Building the network.*** We build the mini-Internet with Docker containers [16]. As opposed to virtual machines, a container does not run its own operating system, but relies on namespaces, a feature available in the Linux kernel. Namespaces isolate software from its environment by partitioning kernel resources. Docker containers are lightweight because they share the host machine's system kernel and computational resources are dynamically allocated. Each component in the mini-Internet (hosts, switches and

routers) runs in its own dedicated Docker container. We then connect the Docker containers following the mini-Internet topology using Open vSwitch [20] bridges and virtual ethernet links. The containers run Debian Stretch [9] and we add the main networking tools (e.g., `traceroute`, `dig`). For the switches, we also use Open vSwitch, a software switch which supports VLANs and the Spanning Tree Protocol. For the routers, we deploy FRRouting [21], an IP routing suite which uses the native Linux/Unix IP networking stack and supports the main routing protocols.

We use OpenVPN [10] to allow the students to virtually connect an external client (e.g., their laptops) into the mini-Internet. The OpenVPN processes run in the server hosting the mini-Internet and are connected to the mini-Internet with virtual links. Each of them listens for new connections on a different port belonging to the host server interface which is connected to the actual Internet. By choosing a specific port, students can therefore decide where to connect in the topology.

***Setting up monitoring and debugging tools.*** For the looking glass, we automatically pull the routing table of each router every minute and upload them to a website. For the measurement platform, we use a dedicated container, connected to every AS and accessible by all the students, from where they can run measurements. Two additional containers are created, one is dedicated for the connectivity matrix and the other one for the DNS service. These two containers are connected to every AS but are not accessible by the students. The container used for the connectivity matrix performs ping measurements at regular intervals between all the pairs of ASes and the results are uploaded to a webpage. For the DNS service, we automatically generate the configuration file and run a `bind9` [11] server in a dedicated container.

***Isolated student access.*** Our students should be able to easily access all their network devices but must not have access to containers of any other group. To achieve that, we rely on the natively provided isolation of Docker containers as well as SSH connections. More precisely, we deploy one additional container for each group of students that we use as a "proxy" and tunnel the incoming SSH connections to the corresponding proxy container based on the port number. We allocate one port number to each group and share the SSH password for a given proxy container only with the students of the corresponding group. From a proxy container, a student can then easily jump into the CLI of one of his or her virtual devices using a simple script that relies on SSH and public/private key pairs automatically generated during the mini-Internet startup process. The following commands illustrate how to access router 3 in AS1 (port 2001 is allocated to the proxy of AS1):

```
> ssh -p 2001 root@server.ethz.ch
g1-proxy> ./goto.sh 3 router    # Could also be "host"
3-router# show ip bgp
```

Observe that the students can setup a key-based SSH authentication to simplify the access to the proxy container.

## 6 ADAPTING THE MINI-INTERNET

This section highlights several ideas on how instructors can increase the authenticity of the mini-Internet infrastructure as well as the difficulty of the questions (e.g., for more advanced classes).

***Adapting the topology and the monitoring tools.*** Instructors can easily tailor the L2, L3 and AS-level topology of the mini-Internet to their requirements. Among others, they can adapt the number of routers within each AS or the number of ASes. A larger number of routers raises interesting scalability questions which opens new possibilities to teach about e.g., hierarchical routing.

To make the mini-Internet even more realistic, instructors can introduce latency on certain links to mimic e.g., the geographical location of certain ASes and IXPs. In addition, instructors could also fail links and/or routers while the project is running allowing students to test their network resiliency. Instructors could furthermore deploy new monitoring tools (such as BMP [23]) or modify the existing ones to only show partial information (e.g., by configuring ICMP filtering). More generally, whatever feature the underlying software tools (e.g., FRRouting or Open vSwitch) support can also be used in the mini-Internet.

***Adapting the questions.*** Besides configuring additional protocols such as IPv6 or MPLS, instructors could also completely change the structure of the questions and the overall teaching goals. To list a few examples, one could confront students with a "black-box" network where they first have to use measurement tools (e.g., `traceroute`) to figure out and visualize the topology of their network as well as the interconnections with other student groups. Another idea relates to the grading methodology. Instead of grading students based on the correctness of their configurations, one can introduce a virtual currency and "bill" the students according to the amount of traffic transiting through their network depending on the business relationships with their neighbors. Yet another idea would be to rely on the mini-Internet to train students or network operators to correctly use and implement emerging technologies such as to validate the origin of BGP routes using the RPKI infrastructure [15].

## 7 EVALUATION

We now show that our platform is well-suited to be used as a practical project in computer network courses with 100+ students.[2] We evaluate it on an Ubuntu 18.04.3 server with 24 Intel Xeon CPU cores @ 2.30 GHz, 256 GB of memory and running the 4.15.0 Linux kernel. We always fully configure hosts, switches and routers and use the 2019 topologies depicted in Fig. 1a and Fig. 1b. For tests with 60 ASes, we use the topology in Fig. 2. For topologies with 20 or 40 ASes we keep the same AS-level structure but reduce the number of regions accordingly e.g., we use two regions to form a mini-Internet with 20 ASes.

***The mini-Internet is easy and relatively fast to setup.*** To start the mini-Internet, the instructor only has to define the topology in the configuration files and run a Bash script. Fig. 4 reports the startup times depending on the number of ASes in the mini-Internet. We can see that for 60 ASes i.e., the size we used in 2019 at ETH Zurich, it takes around 12 hours to build the mini-Internet. This

[2] In 2020, we successfully used the mini-Internet for 150 students.

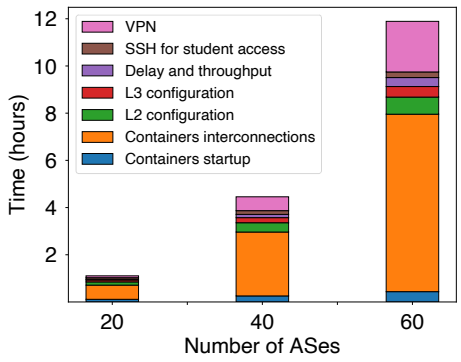
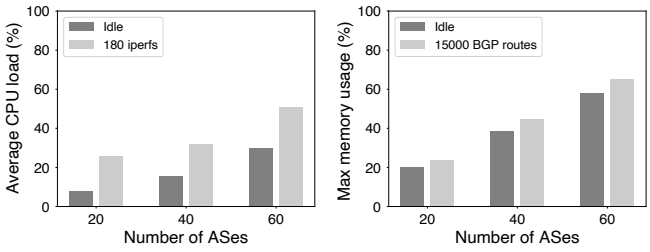



**Figure 4: Startup time vs the number of ASes.**




**(a) CPU load** **(b) Memory usage**

**Figure 5: The CPU (Fig. 5a) and memory (Fig. 5b) used by a mini-Internet with 60 ASes when idle and under load.**

is acceptable given that this step is done automatically and only once at the beginning of the semester. Digging deeper, we see that the interconnection of the containers with Open vSwitches and virtual links has the longest setup time. With 60 ASes it takes 7.5 hours to create the 497 Open vSwitches and the 7191 virtual links used to connect the 1690 containers. Similarly, enabling the VPN service also takes time as it needs a lot of virtual links and we have to generate a set of keys and certificates, one for each VPN server.

***One server is enough for 100 students.*** Fig. 5 depicts the CPU and memory usage as a function of the size of the mini-Internet. In the idle state i.e., the mini-Internet is fully started but no traffic is being forwarded, the average CPU load is 29.7% and up to 58% of the memory is used (topology with 60 ASes). To simulate the network under load, we perform two tests. First, we start 180 ten-minutes `iperf` sessions between random pairs of hosts. In the case of 60 ASes, each student group therefore simultaneously sends traffic to three other groups on average. Second, we also measure the load when we advertise a high number of BGP prefixes in the entire mini-Internet. This test is based on the observation that students often advertise more prefixes than expected. For example, some groups advertise every single used /24 prefix instead of only their /8 prefix. Therefore we measure the load after advertising 15000 prefixes (250 distinct BGP prefixes per group with 60 ASes).

Because all the virtual links are bandwidth limited, the `iperf` sessions do not overload the server. The CPU load only increases to 51% with 60 ASes (see Fig. 5a) whereas the effect on the memory is negligible. The 15000 BGP routes lead to a high CPU load (>80%) during the convergence time, and an additional memory usage over time (up to 65.2% with 60 ASes, see Fig. 5b). The results thus indicate that one server can easily handle a mini-Internet with 60 ASes, enough for 108 students if we allocate three students to each transit AS (see §3). Although we never had issues during the last four years, we note that a malicious student group could probably overload the server and impact part of the mini-Internet (e.g., by advertising hundreds of thousands of fake BGP routes). Yet, we mitigate the potential impact by periodically and automatically saving the configuration files of each router and switch in the network. Therefore the student's progress is not lost should we have to restart one or multiple containers. In addition, we could also maintain logs to detect malicious activities.

## 8 CONCLUSION AND FUTURE WORK

We propose to teach students how the Internet *practically* works by having an entire classroom build and operate their own Internet infrastructure. We describe the design and the implementation of a platform that can support this and make it publicly available. Our four year-long experience running the project at ETH Zurich tells us that not only do the students like the project, they also gain a much deeper understanding of the various Internet mechanisms.

We have been nurturing the project over four years and intend to continue to do so. Below is a list of improvements we plan to do.

***Support for multi-server deployments.*** In some classes the number of students can easily exceed 100, in which case a single server might not have enough resources to run the entire mini-Internet. We intend to soon support the ability to run the mini-Internet over multiple servers, in a transparent manner.

***Auto-grading.*** Manually grading the students by carefully checking the configuration files of their routers and switches is time-consuming. Here, we intend to develop tools to parse their configuration files and actively send traffic through the network to automatically check the correctness of the configurations and verify the implemented policies at runtime.

***Connecting the real Internet to the mini-Internet.*** We plan to connect the real Internet to the mini-Internet to enable students to browse the web or watch videos from a host inside the mini-Internet. This must be done carefully as not all the Internet prefixes can be advertised in the mini-Internet (some prefixes are already allocated in the mini-Internet itself), and the additional load must be tightly controlled (traffic volume and number of prefixes).

## ACKNOWLEDGEMENTS

We thank all the teaching assistants from the Networked Systems Group at ETH Zurich that were involved in this project since 2016. We are also grateful to the SIGCOMM CCR anonymous reviewers for their insightful comments. We would also like to thank Ethan Katz-Bassett and Oliver Hohlfeld for their insightful feedback, and for having used *and* improved the platform themselves.

This work was partially supported by a Swiss National Science Foundation Grant (Data-Driven Internet Routing, #200021-175525).

## REFERENCES


[1] BGP Looking Glasses for IPv4/IPv6, Traceroute & BGP Route Servers. 2019. https://www.bgp4.as/looking-glasses.
[2] The mini-Internet project. http://mini-inter.net/.
[3] Florian Baumgartner, Torsten Braun, Eveline Kurt, and Attila Weyland. Virtual Routers: A Tool for Networking Research and Education. *ACM CCR 2003.*
[4] Massimo Candela. TraceMON: Network Debugging Made Easy. 2017. https://labs.ripe.net/Members/massimo_candela/tracemon-traceroute-visualisation-network-debugging-tool.
[5] R. I. Dinita, G. Wilson, A. Winckles, M. Cirstea, and A. Jones. A cloud-based virtual computing laboratory for teaching computer networks. In *13th International Conference on Optimization of Electrical and Electronic Equipment. 2012.*
[6] Dalibor Dobrilovic and Borislav Lj. Odadzic. Virtualization Technology as a Tool for Teaching Computer Networks. 2008.
[7] Lixin Gao and Jennifer Rexford. Stable Internet Routing Without Global Coordination. *IEEE/ACM ToN 2001.*
[8] Chuanxiong Guo and al. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *ACM SIGCOMM 2015.*
[9] Docker Official Images. Debian Stretch. https://hub.docker.com/_/debian/.
[10] OpenVPN Inc. OpenVPN. https://openvpn.net.
[11] Internet Systems Consortium, Inc. Bind9. Versatile, classic, complete name server software. https://www.isc.org/bind/.
[12] Internet2. Network Infrastructure Topology. https://www.internet2.edu/media/medialibrary/2017/09/25/I2-Network-Infrastructure-Topology-Alllogos-201705_hr8gwSg.pdf.
[13] James F Kurose. *Computer networking: A top-down approach featuring the internet.* Pearson Education, 2005.
[14] Bob Lantz, Brandon Heller, and Nick McKeown. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *ACM HotNets 2010.*
[15] Matt Lepinski and Stephen Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480. https://rfc-editor.org/rfc/rfc6480.txt
[16] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J. 2014.*
[17] RIPE NCC. RIPE Atlas. https://atlas.ripe.net.
[18] Juniper Networks. Whats Behind Network Downtime? Proactive Steps to Reduce Human Error and Improve Availability of Networks. Technical Report. 2008.
[19] Larry L Peterson and Bruce S Davie. *Computer networks: a systems approach.* Elsevier.
[20] Ben Pfaff and al. The Design and Implementation of Open vSwitch. In *NSDI 2015.*
[21] FRRouting Project. FRRouting. https://frrouting.org.
[22] A. Ruiz-Martínez and al. Teaching Advanced Concepts in Computer Networks: VNUML-UM Virtualization Tool. *IEEE Transactions on Learning Technologies. 2013.*
[23] John Scudder, Rex Fernando, and Stephen Stuart. BGP Monitoring Protocol (BMP). RFC 7854. https://rfc-editor.org/rfc/rfc7854.txt
[24] SWITCH. The SWITCHlan backbone. https://www.switch.ch/network/infrastructure/backbone/.
[25] ETH Zurich. Communication Networks. https://comm-net.ethz.ch/.