

An Open Platform to Teach How the Internet Practically Works

Thomas Holterbach &
Tobias Bühler

SIGCOMM Best of CCR

12th August 2020

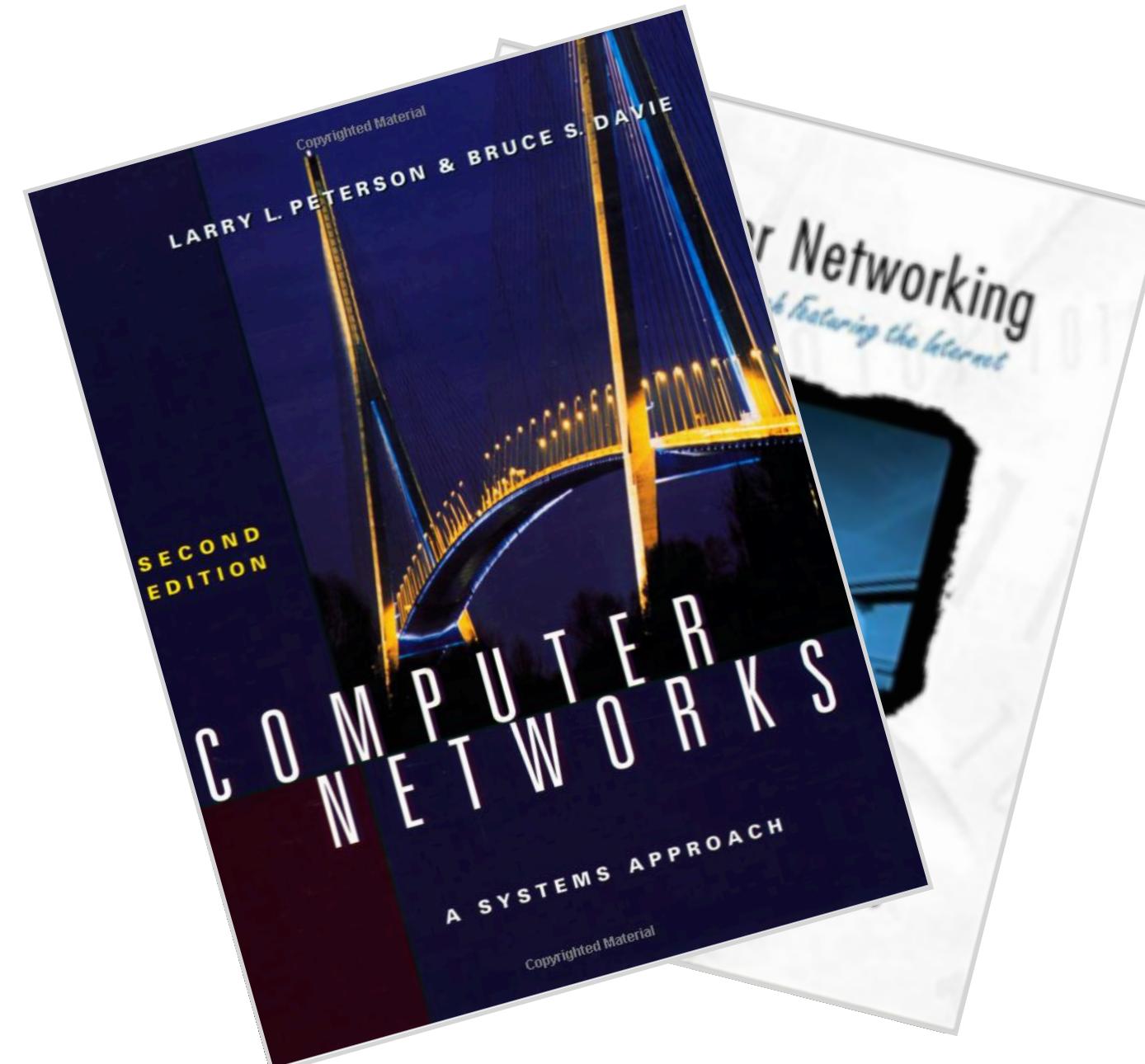
Joint work with Tino Rellstab
and Laurent Vanbever

ETH zürich

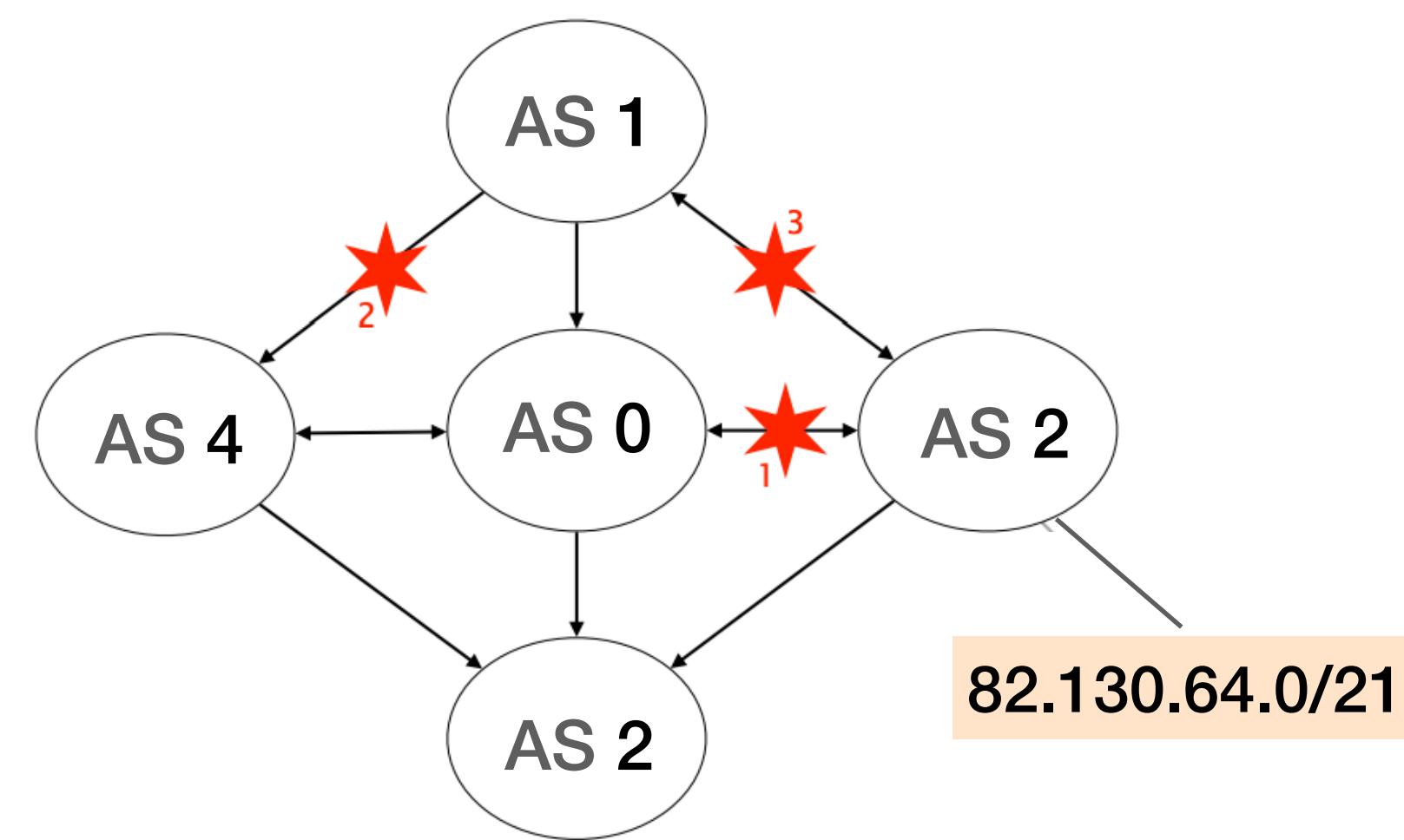
How do we traditionally teach how the Internet works?

How do we traditionally teach how the Internet works?

theory



exercises



Which messages are exchanged?

labs



These concepts are not sufficient to understand
how the Internet *practically* works

In practice, there are peering agreements with stringent SLAs



Network operators talking
during NANOG'76

In practice, there are **thousands** of ASes and connectivity must be monitored **network-wide**



IXP Country Jedi
Emile Aben

In practice, debugging can be **tricky**

Anybody else is experiencing packet loss since last Tuesday across the AT&T network in the L.A. area?
I'm seeing it coming from both Zayo and HE

8. ae2.cs1.lga5.us.zip.zayo.com
9. ae18.ter1.lga5.us.zip.zayo.com
10. 192.205.36.105
11. cr1.n54ny.ip.att.net
12. cgcil22crs.ip.att.net
13. cgcil21crs.ip.att.net
14. dvmco22crs.ip.att.net
15. slkut21crs.ip.att.net
16. la2ca21crs.ip.att.net
17. gar20.la2ca.ip.att.net

NANOG mailing list
December 9, 2019

At ETH Zurich, we let the students operate their own **mini-Internet**,
altogether, like if they were the network operators



ETH students working on the mini-Internet

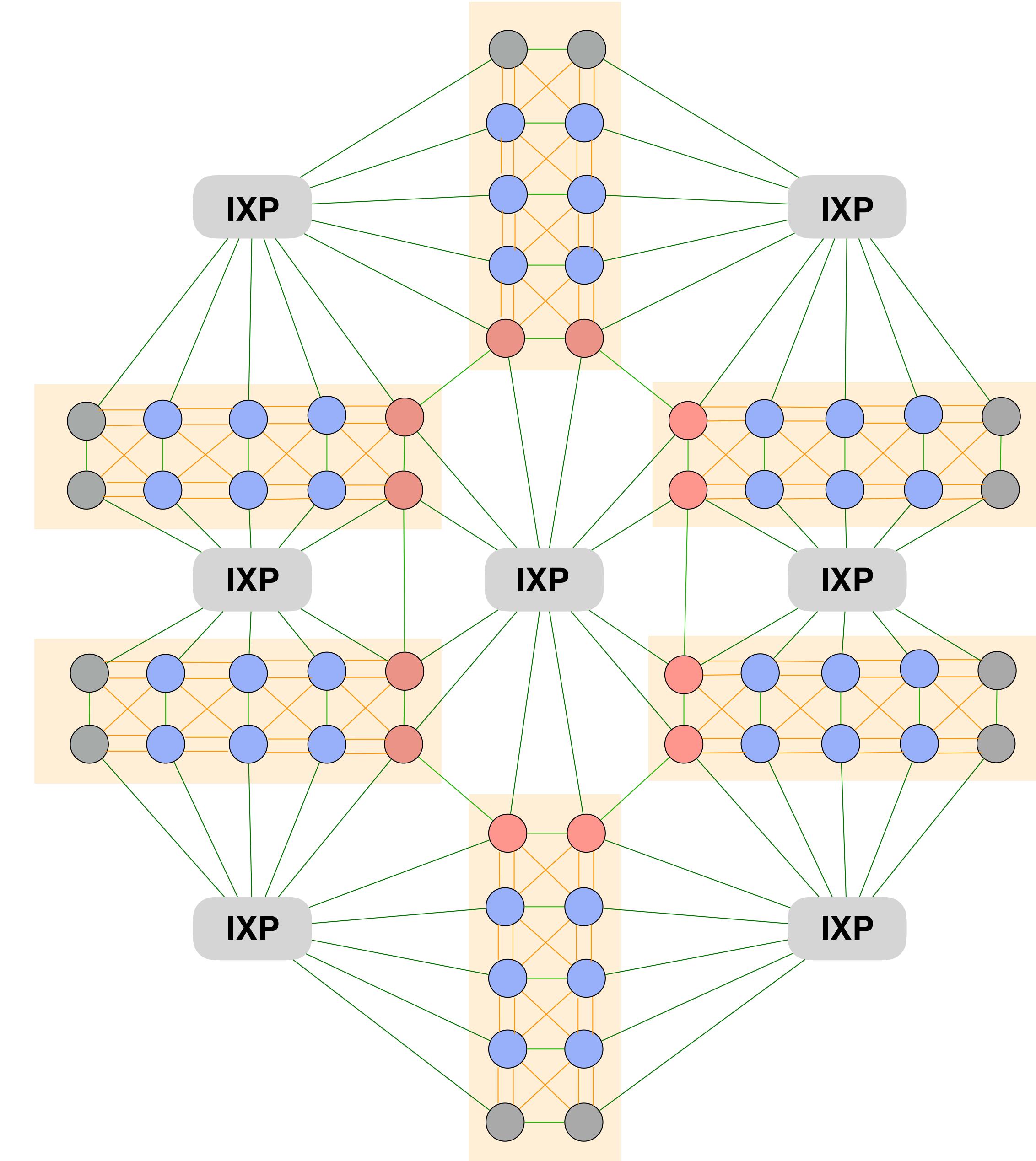
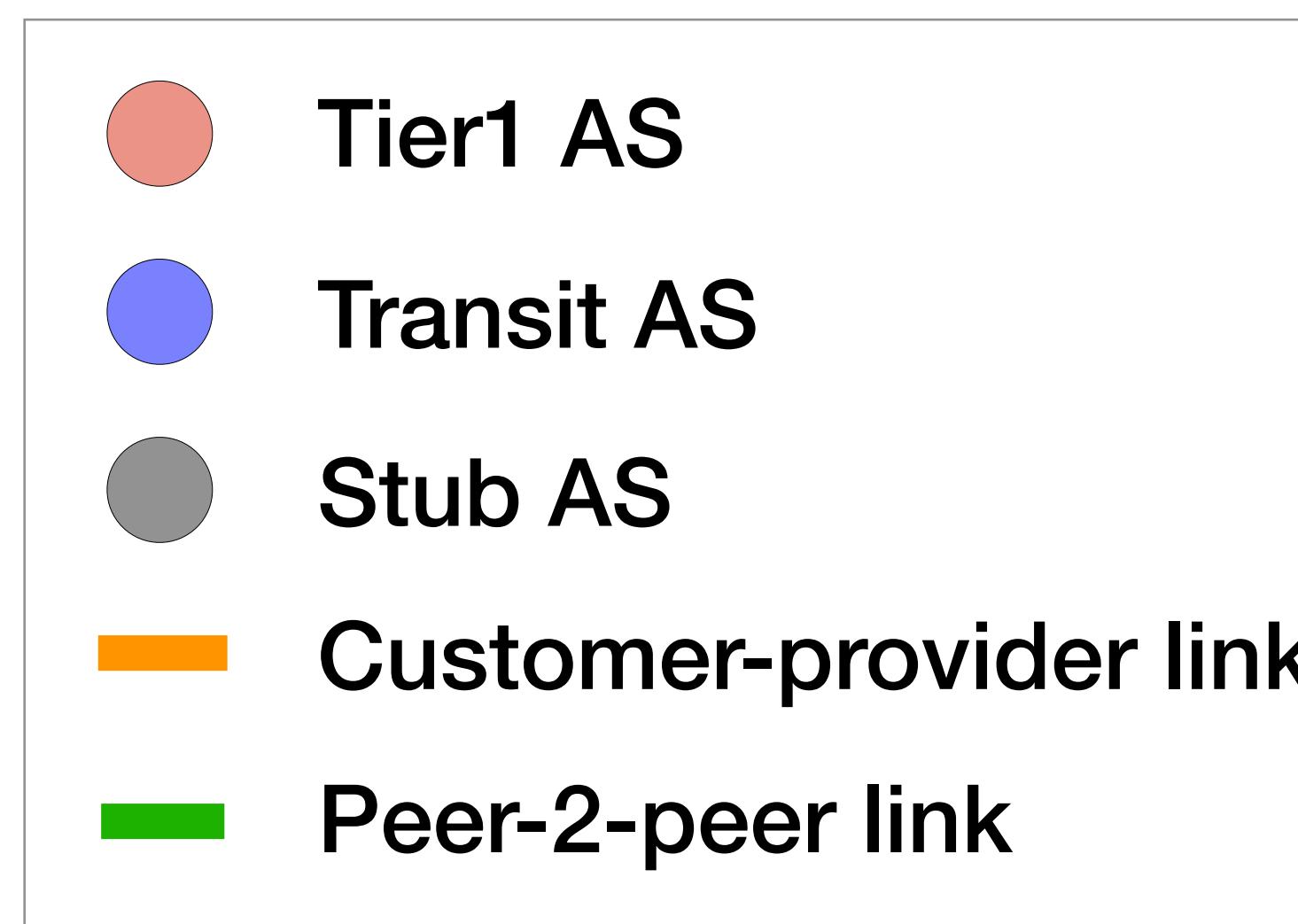
Outline

1. The mini-Internet **mimics** the real one and is entirely **virtual**
2. The mini-Internet turns the students into **network operators**
3. The mini-Internet provides students with tools to **ease operations**

Outline

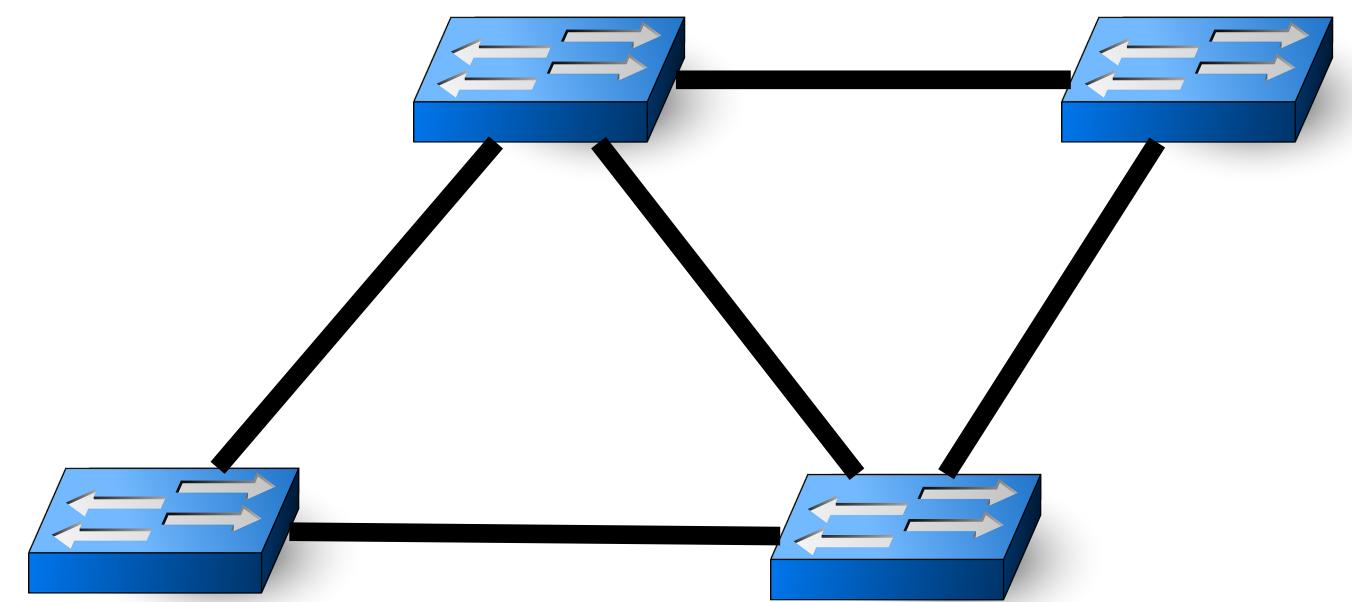
1. The mini-Internet **mimics** the real one and is entirely **virtual**
2. The mini-Internet turns the students into **network operators**
3. The mini-Internet provides students with tools to **ease operations**

The AS-level topology we use in our mini-Internet



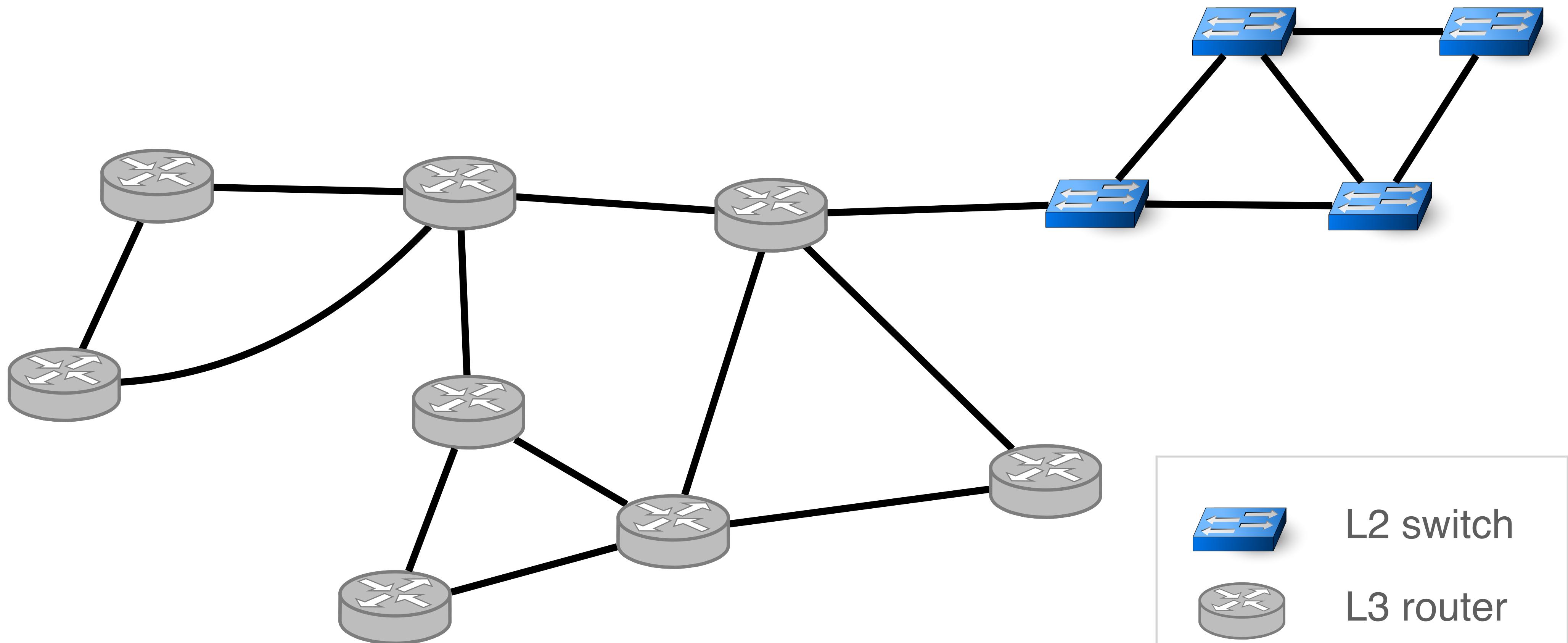
**We build internal topologies
with the technologies used in practice**

We build internal topologies
with the technologies used in practice

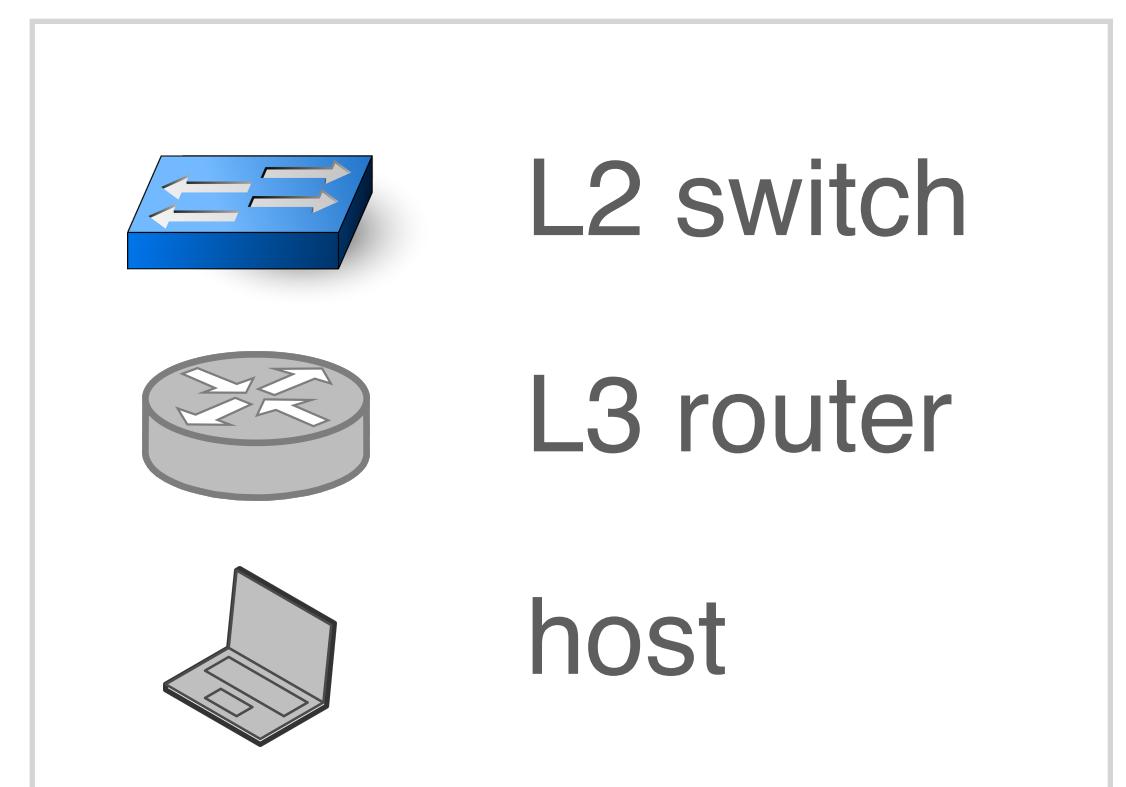
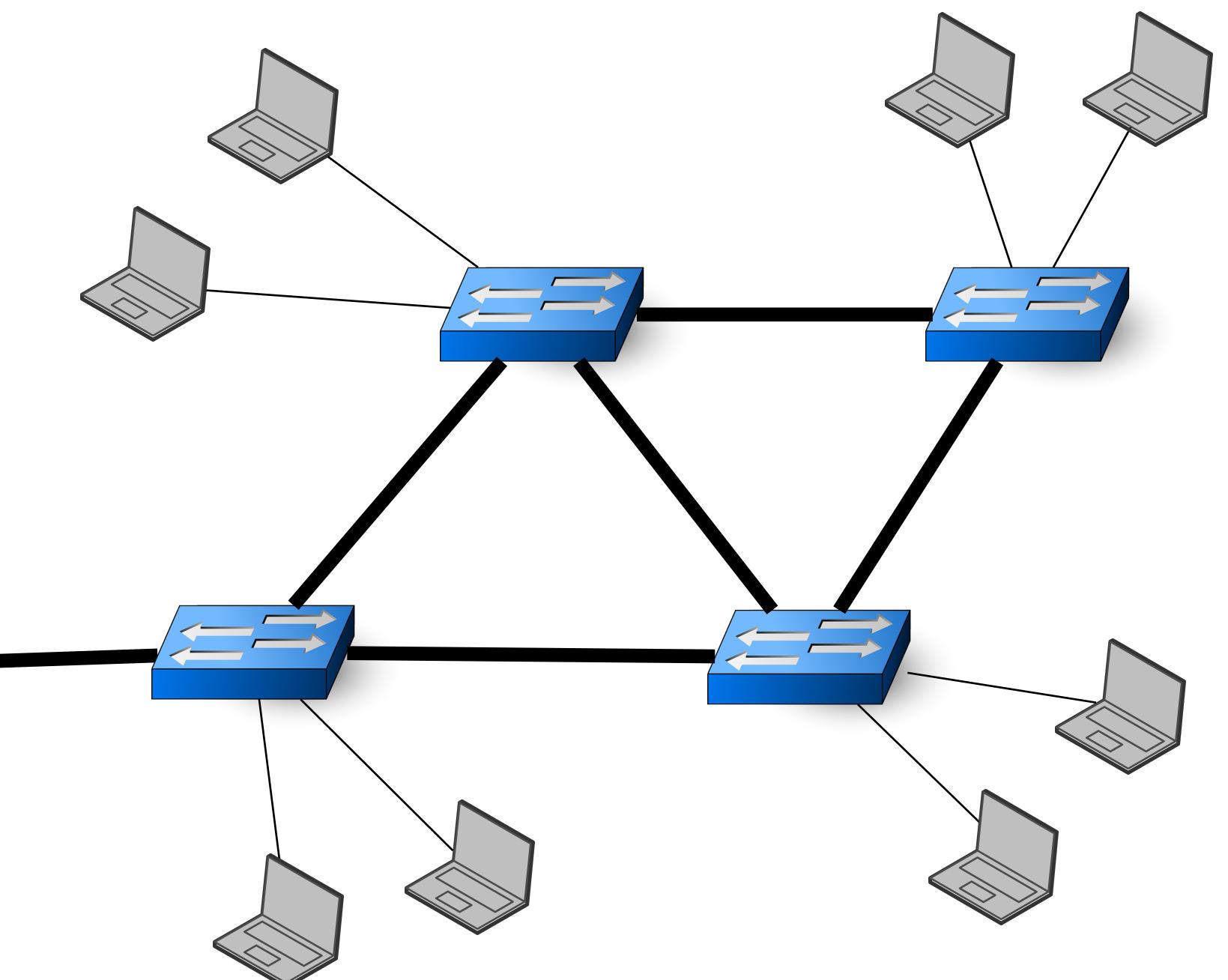
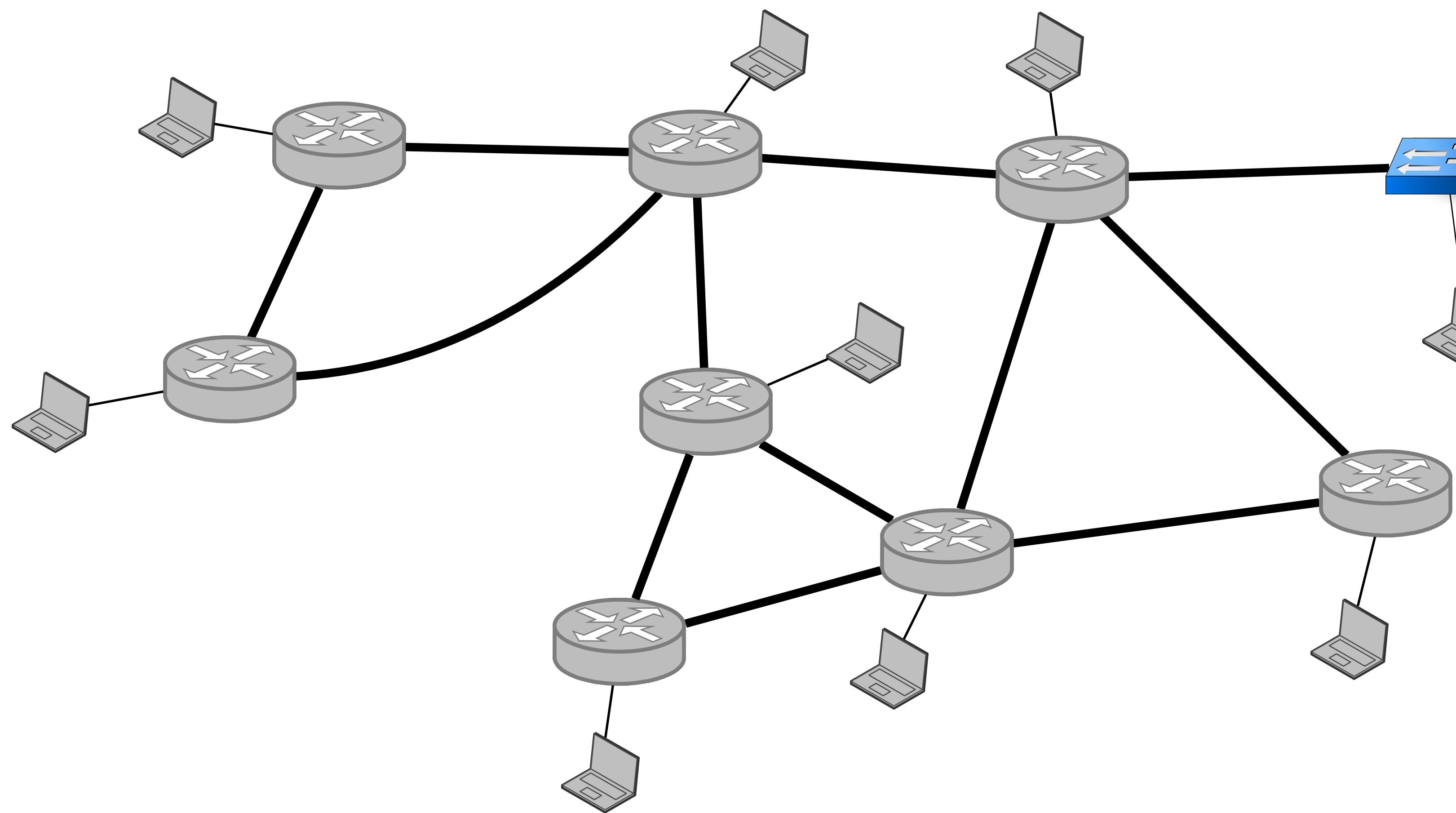


L2 switch

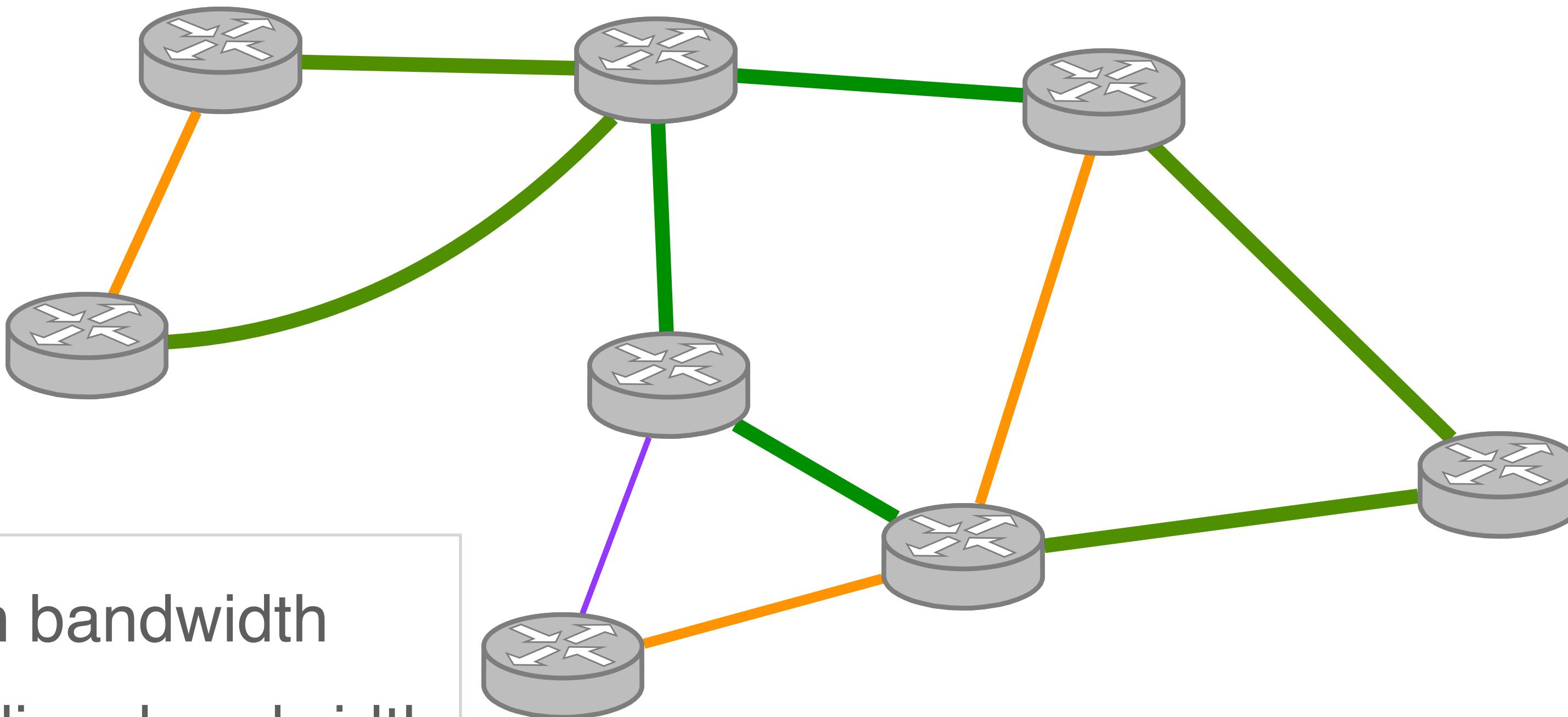
We build internal topologies
with the technologies used in practice



We build internal topologies
with the technologies used in practice

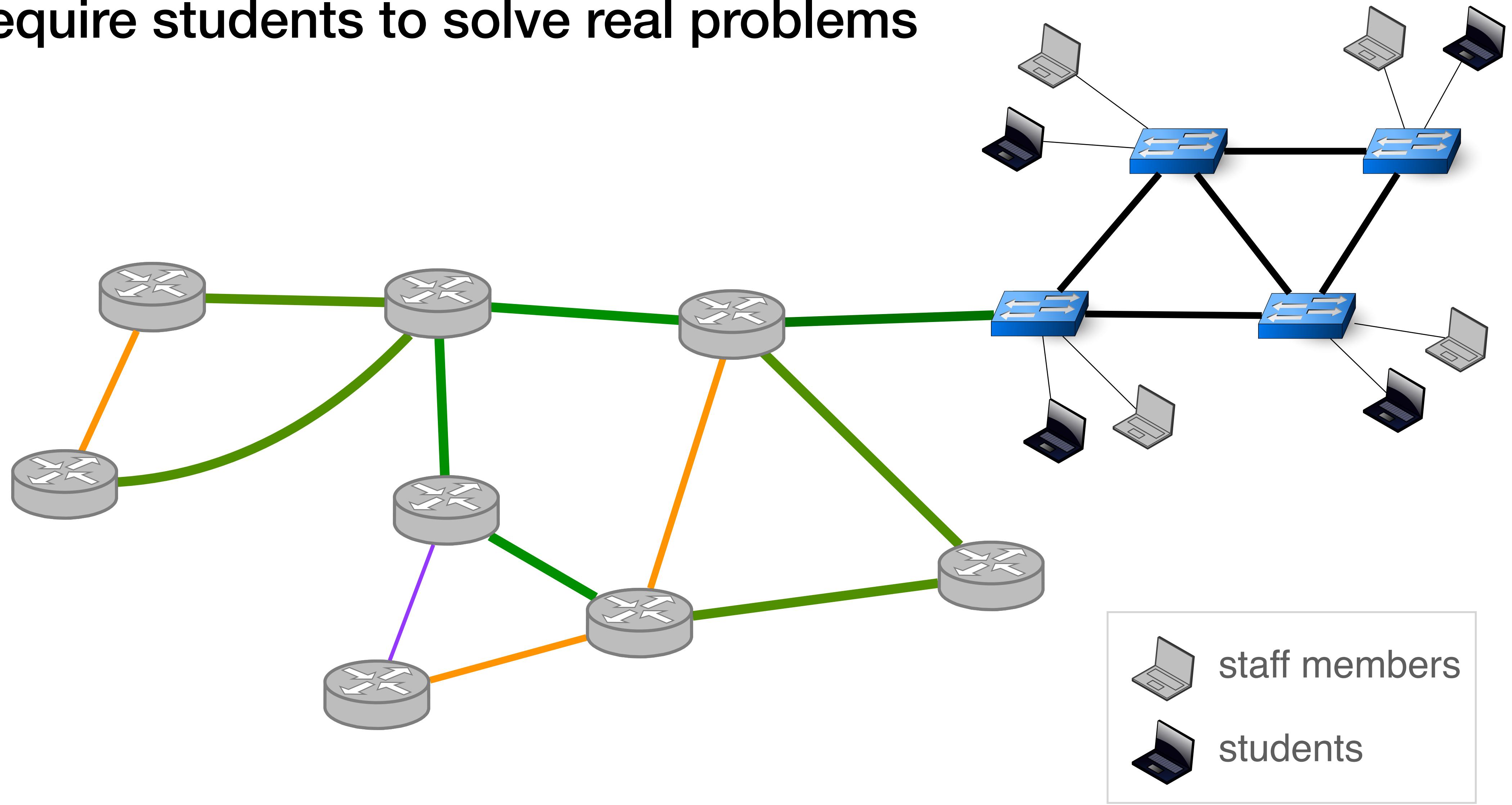


We build realistic internal topologies
that require students to solve real problems

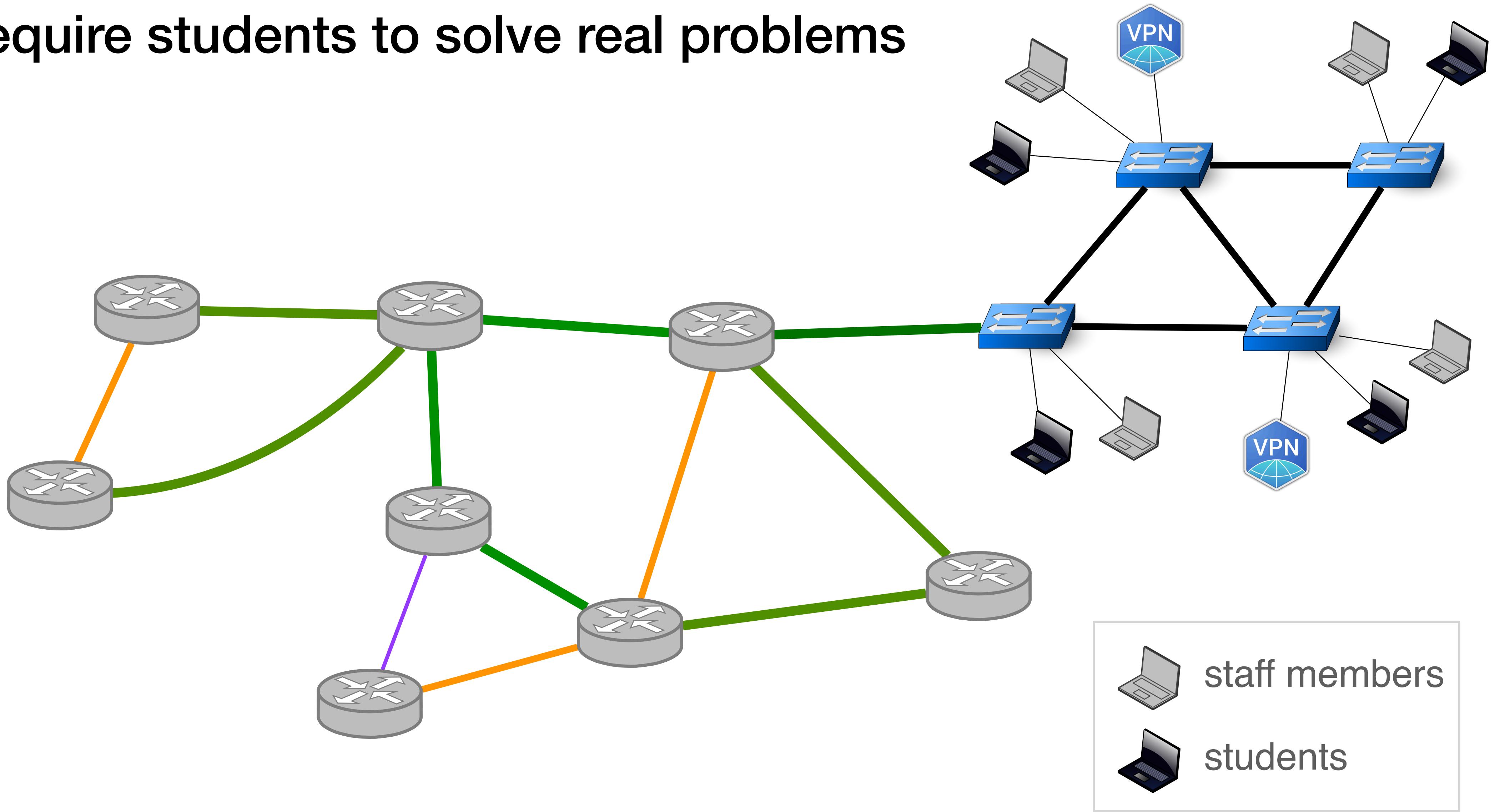


- high bandwidth
- medium bandwidth
- low bandwidth

We build realistic internal topologies
that require students to solve real problems

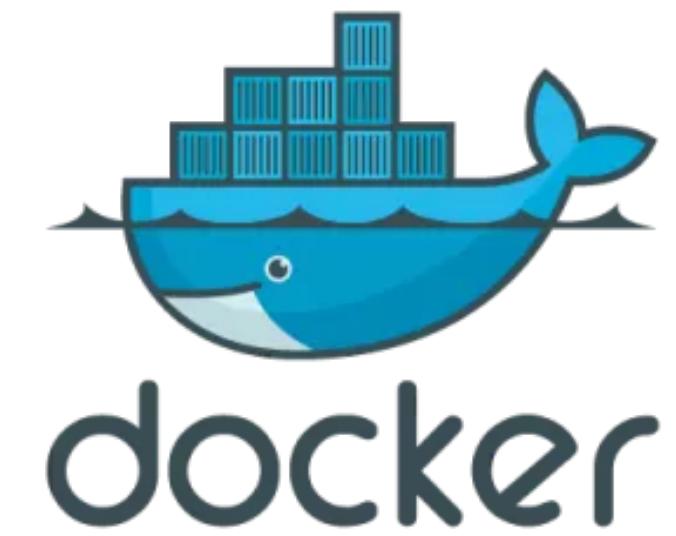


We build realistic internal topologies
that require students to solve real problems



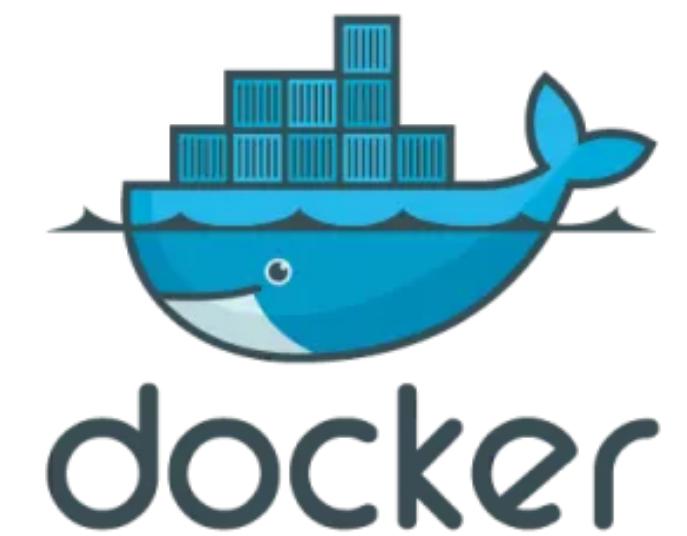
The mini-Internet runs in a **single** server

Each component (router, switch and host)
runs in its own docker container



The mini-Internet runs in a **single** server

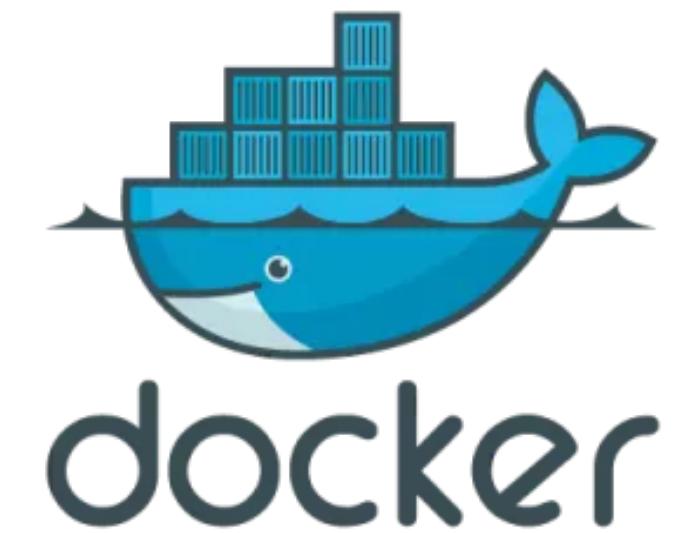
Each component (router, switch and host)
runs in its own docker container



With connect the containers following
the topology using virtual links

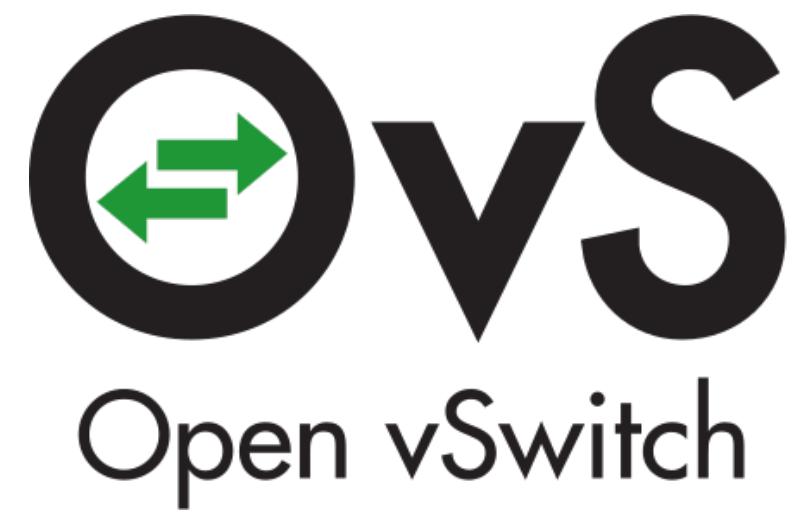
The mini-Internet runs in a **single** server

Each component (router, switch and host)
runs in its own docker container



With connect the containers following
the topology using virtual links

We use the state of the art software suite
for the routers and switches



Outline

1. The mini-Internet **mimics** the real one and is entirely **virtual**
2. The mini-Internet turns the students into **network operators**
3. The mini-Internet provides students with tools to **ease operations**

We give one transit AS and one IP prefix to each group of students

We give one transit AS and one IP prefix to each group of students

Goal: enabling Internet-wide connectivity

**Students have to enable internal connectivity
and perform traffic engineering**

In the L2 network
e.g., custom spanning tree and VLANs

In the L3 network
e.g., load-balancing

We organise a Hackathon where students gather to configure BGP sessions



mini-Internet Hackathon, April 19, 2018

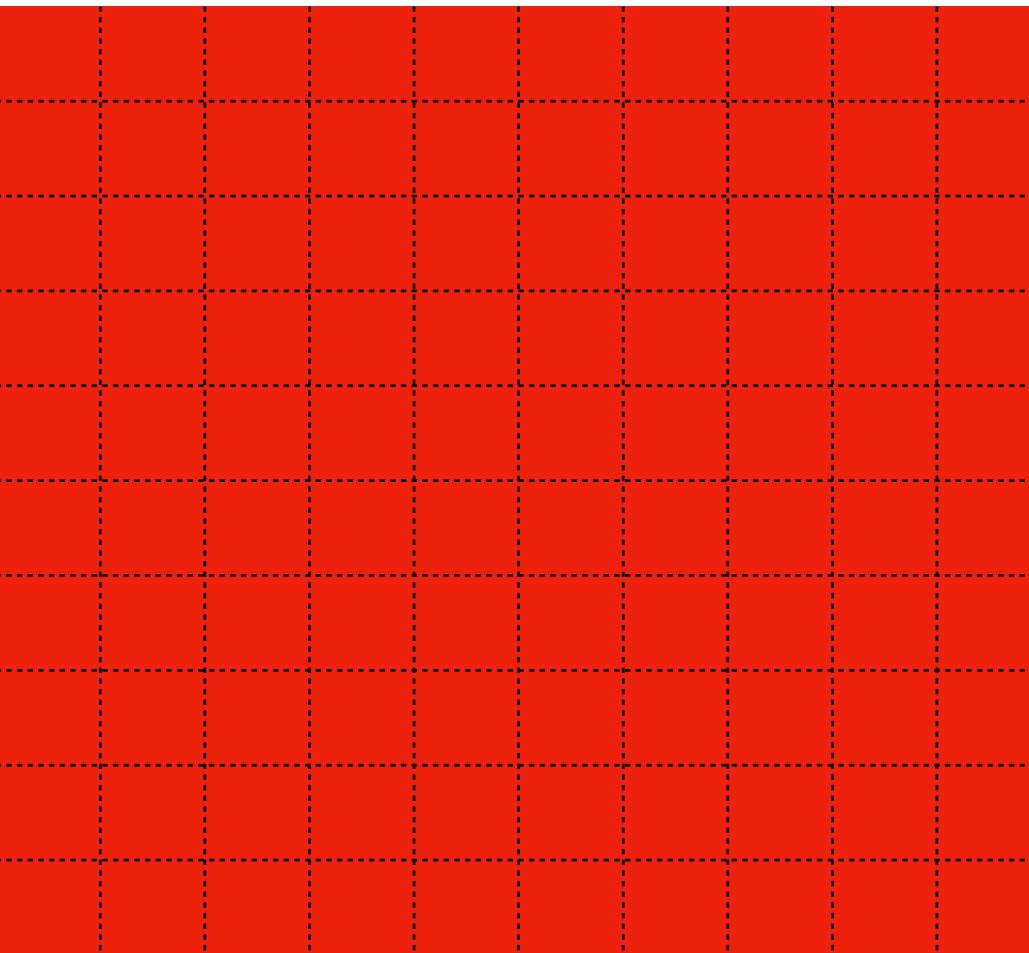
**Besides enabling BGP sessions,
students have to implement **routing policies****

Following business agreements
e.g., local-preference and exportation rules

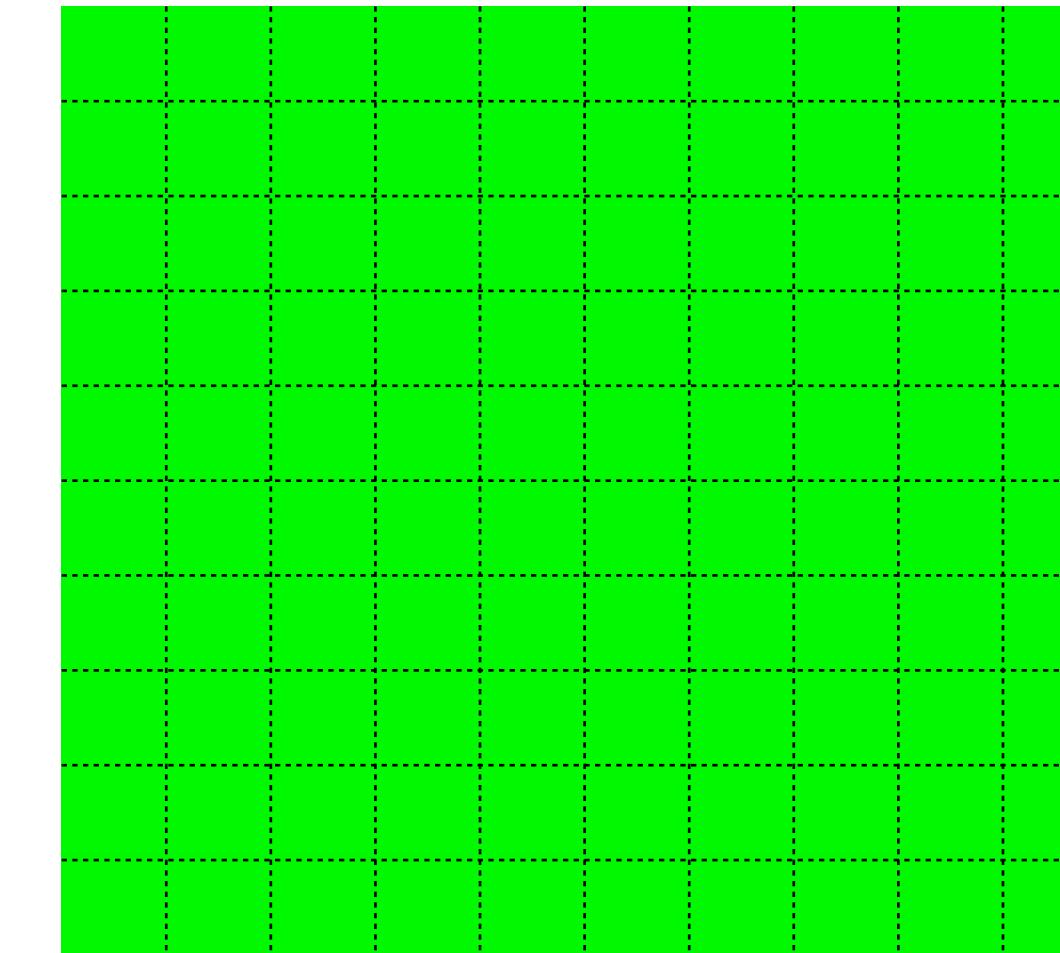
Following preferences
e.g., one provider is preferred

**This year, at the end of the project the mini-Internet was
fully connected**

At the beginning



This year, at the end



Device access

Students access a so call proxy container

From there they can access every device with a single command:

`./goto.sh <device_name> <device_type>`

To access a router called NEWY:

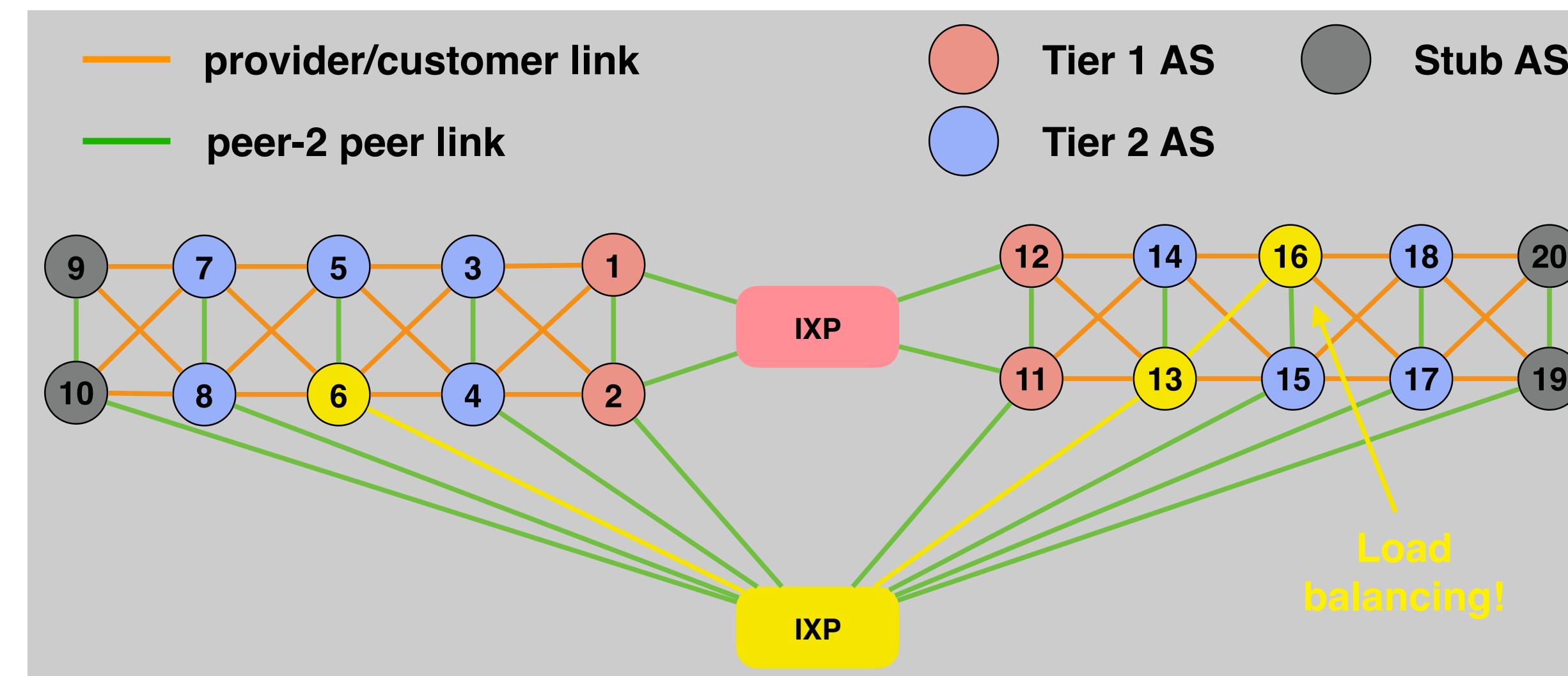
`./goto.sh NEWY router`

Let's perform a traceroute from a host

```
root@NEWY_host:~# traceroute 16.103.0.1
traceroute to 16.103.0.1 (16.103.0.1), 30 hops max, 60 byte packets
 1 NEWY-host.group6 (6.105.0.2)  0.427 ms  0.107 ms  0.030 ms
 2 180.82.0.13 (180.82.0.13)  8.505 ms  7.949 ms  7.898 ms
 3 ATLA-NEWY.group13 (13.0.11.2)  7.829 ms  7.714 ms  7.660 ms
 4 ZURI-LOND.group16 (16.0.2.1)  12.675 ms  12.620 ms ZURI-PARI.group16 (16.0.1.1)  12.560 ms
 5 PARI-NEWY.group16 (16.0.5.1)  12.234 ms  12.066 ms  11.990 ms
 6 host-PARI.group16 (16.103.0.1)  12.053 ms  9.375 ms  9.281 ms
root@NEWY_host:~# █
```

Let's perform a traceroute from a host

```
root@NEWY_host:~# traceroute 16.103.0.1
traceroute to 16.103.0.1 (16.103.0.1), 30 hops max, 60 byte packets
 1 NEWY-host.group6 (6.105.0.2)  0.427 ms  0.107 ms  0.030 ms
 2 180.82.0.13 (180.82.0.13)  8.505 ms  7.949 ms  7.898 ms
 3 ATLA-NEWY.group13 (13.0.11.2)  7.829 ms  7.714 ms  7.660 ms
 4 ZURI-LOND.group16 (16.0.2.1)  12.675 ms  12.620 ms ZURI-PARI.group16 (16.0.1.1)  12.560 ms
 5 PARI-NEWY.group16 (16.0.5.1)  12.234 ms  12.066 ms  11.990 ms
 6 host-PARI.group16 (16.103.0.1)  12.053 ms  9.375 ms  9.281 ms
root@NEWY_host:~# █
```



**Even a single traceroute command
can reveal a lot of important events**

**Students can confirm their forwarding policies
as well as the correct usage of IXPs**

**They observe load-balancing
which at first often looks confusing in the traceroute output**

**We often observe that students will start to contact other groups/ASes
for example if they observe a mistake in their policies**

Outline

1. The mini-Internet **mimics** the real one and is entirely **virtual**
2. The mini-Internet turns the students into **network operators**
3. The mini-Internet provides students with tools to **ease operations**

Our students have no a priori knowledge and a limited time budget

Our students have no a priori knowledge and a limited time budget

We **assist** them

Our students have no a priori knowledge and a limited time budget

We **assist** them

We organise Q&A sessions every week
where teaching assistants provide help

Our students have no a priori knowledge and a limited time budget

We **assist** them

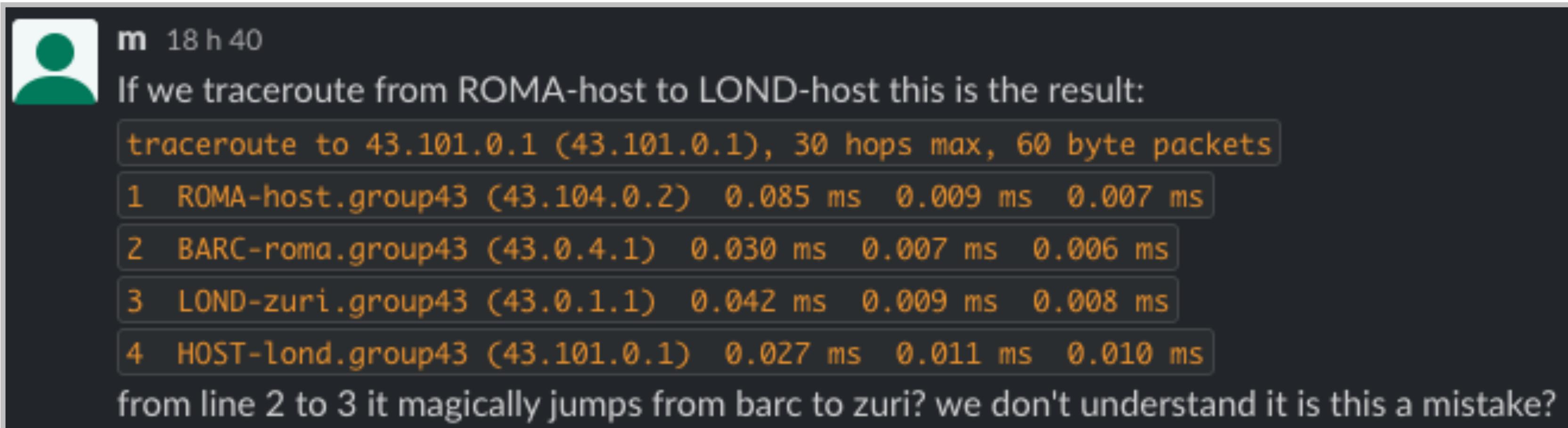
We organise Q&A sessions every week
where teaching assistants provide help

We use a dedicated Slack channel
where students can ask questions any time

Our students have no a priori knowledge and a limited time budget
We **assist** them

We organise Q&A sessions every week
where teaching assistants provide help

We use a dedicated Slack channel
where students can ask questions any time

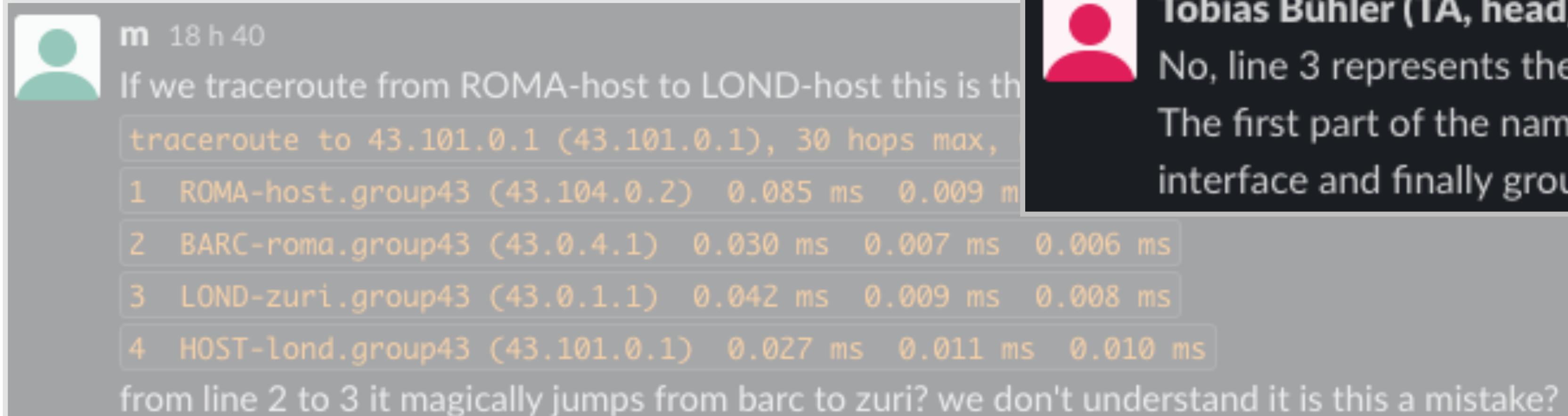


m 18 h 40
If we traceroute from ROMA-host to LOND-host this is the result:
`traceroute to 43.101.0.1 (43.101.0.1), 30 hops max, 60 byte packets
1 ROMA-host.group43 (43.104.0.2) 0.085 ms 0.009 ms 0.007 ms
2 BARC-roma.group43 (43.0.4.1) 0.030 ms 0.007 ms 0.006 ms
3 LOND-zuri.group43 (43.0.1.1) 0.042 ms 0.009 ms 0.008 ms
4 HOST-lond.group43 (43.101.0.1) 0.027 ms 0.011 ms 0.010 ms`
from line 2 to 3 it magically jumps from barc to zuri? we don't understand it is this a mistake?

Our students have no a priori knowledge and a limited time budget
We **assist** them

We organise Q&A sessions every week
where teaching assistants provide help

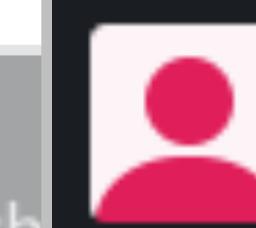
We use a dedicated Slack channel
where students can ask questions any time



m 18 h 40
If we traceroute from ROMA-host to LOND-host this is the traceroute to 43.101.0.1 (43.101.0.1), 30 hops max,

1 ROMA-host.group43 (43.104.0.2) 0.085 ms 0.009 ms
2 BARC-roma.group43 (43.0.4.1) 0.030 ms 0.007 ms 0.006 ms
3 LOND-zuri.group43 (43.0.1.1) 0.042 ms 0.009 ms 0.008 ms
4 HOST-lond.group43 (43.101.0.1) 0.027 ms 0.011 ms 0.010 ms

from line 2 to 3 it magically jumps from barc to zuri? we don't understand it is this a mistake?



Tobias Bühler (TA, head) il y a 10 mois

No, line 3 represents the zuri interface on the LOND router.
The first part of the name indicates the router/host, then interface and finally group.

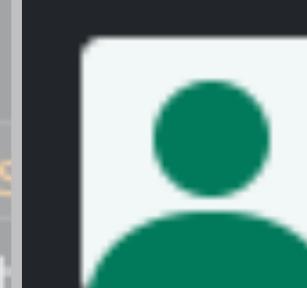
Our students have no a priori knowledge and a limited time budget
We **assist** them

We organise Q&A sessions every week
where teaching assistants provide help

We use a dedicated Slack channel
where students can ask questions any time

 m 18 h 40
If we traceroute from ROMA-host to LOND-host this is the
traceroute to 43.101.0.1 (43.101.0.1), 30 hops max,
1 ROMA-host.group43 (43.104.0.2) 0.085 ms 0.009 ms
2 BARC-roma.group43 (43.0.4.1) 0.030 ms 0.007 ms 0.006 ms
3 LOND-zuri.group43 (43.0.1.1) 0.042 ms 0.009 ms 0.008 ms
4 HOST-lond.group43 (43.101.0.1) 0.027 ms 0.011 ms 0.010 ms
from line 2 to 3 it magically jumps from barc to zuri? we don't understand

 Tobias Bühler (TA, head) il y a 10 mois
No, line 3 represents the zuri interface on the LOND router.
The first part of the name indicates the router/host, then
interface and finally group.

 m il y a 10 mois
ok thank you:)

Monitoring and debugging a network is tricky

Monitoring and debugging a network is tricky

We provide **monitoring and debugging tools**

Looking glass: the routing table of every router is available on a web interface

Active probing: the students can run ping and traceroute between any pair of ASes to test connectivity

DNS: the students can use domain names instead of IP addresses

Students are not familiar with routers and switches' CLI

Students are not familiar with routers and switches' CLI

We provide a **documentation** tailored for the mini-Internet



Spring 2018 Prof. L. Vanbever / T. Bühler, R. Birkner, T. Holterbach, R. Meier

Communication Networks

Project 1: Build your own Internet
Deadline: May 3 2018 at 11.59pm

In this document, we first introduce in §1 a set of commands you may need to configure an Open vSwitch. We then show in §2 how to configure a Quagga router.

1 Configuring Open vSwitch

Open vSwitch¹ [1] is one of the most popular software switches. It can typically be used in virtual environments, for instance to connect two virtual machines. When an Open vSwitch is running, a set of commands are available to check its state and configure it. To print a brief overview of the switch state and its parameters, you can use the following command:

```
> ovs-vsctl show
```

This command also tells you the VLANs each port belongs to. One port has the name of the switch and has the type *internal*. This is a local port used by the host to communicate with the switch. You do **not** need to use this port. To get more precise information about the status of the ports, you can use the following command:

```
> ovs-ofctl show NAME
```

↳ NAME is the name of the switch (e.g. ETH1-PC1). To get the current connection and

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

The screenshot shows the GitHub repository page for `nsg-ethz/mini_internet_project`. The repository has 16 stars, 48 forks, and 7 open issues. It contains 2 branches and 0 tags. The `Code` tab is selected, showing a list of recent commits:

Commit	Description	Date
KTrel fix dockerfile for the host and router docker images, and fix one ...	typo assignment folder	4 days ago
2019_assignment_eth	Create README.md	4 months ago
2020_assignment_eth	fix dockerfile for the host and router docker images, and fix on...	5 months ago
platform	added .gitignore file	4 days ago
.gitignore	Create LICENSE	8 months ago
LICENSE	Update README.md	26 days ago
README.md		

The `README.md` file contains the following content:

An Open Platform to Teach How the Internet Practically Works

Welcome in the official repository of the mini-Internet project.

The mini-Internet project

A mini-Internet is a virtual network mimicking the real Internet. Among others, there are routers, switches and hosts that are located in different ASes. A mini-Internet runs in a single server and is tailored to teach how the Internet practically works. Each components of the network is running in its own dedicated linux container, that are remotely accessible by the students with simple ssh connections.

The mini-Internet project is the flagship piece of our [Communication Networks course](#) at ETH Zurich since 2016. The concept is rather simple: we let each student group operate their own AS. Their goal? Enabling Internet-wide connectivity.

About: The official repository of the mini-Internet exercise.

Packages: No packages published. [Publish your first package](#).

Contributors: KTrel (4), buehlert, rsanger, noah95.

Languages: Shell 68.2%, Perl 18.9%, Python 9.8%, Dockerfile 2.9%.

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

Prerequisite

To build the mini-Internet, you need to install the following software on the server which hosts the mini-Internet.

Install the Docker Engine

To run all the different components in the mini-Internet (hosts, switches, routers, ...) we use Docker containers.

Follow this [installation guide](#) to install docker. In the directory `docker_images` you can find all the Dockerfile and docker-start files used to build the containers. In case you want to add some functionalities into some of the docker containers, you can update these files and build your own docker images:

```
docker build --tag=your_tag your_dir/
```

Then, you have to manually update the scripts in the `setup` directory and run your custom docker images instead of the ones we provide by default.

Install Open vSwitch

We use the Open vSwitch framework in two ways: (i) to build the L2 component of the mini-Internet and (ii) to connect Docker containers together.

```
sudo apt-get install openvswitch-switch
```

For further information, see the [installation guide](#).

Install OpenVPN

Finally, we also need Open VPN which allows the students to connect their own devices to the mini-Internet.

```
sudo apt-get install openvpn
```

Install OpenSSL

Make sure to use [OpenSSL 1.1.1](#) (2018-Sep-11). If you want to use the latest OpenSSL version, then you need to use DH keys of size 2048 (see [here](#)), but that will increase the startup time.

Build the mini-Internet

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

For example the internal links in:
config/internal_links_config.txt

ZURI	PARI	100000	100
ZURI	LOND	100000	1000

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

For example the internal links in:
config/internal_links_config.txt

```
ZURI PARI 100000 100
ZURI LOND 100000 1000
```

Or external connectivity in:
config/external_links_config.txt

```
1 ZURI Peer 2 ZURI Peer 100000 1000 3.0.1.0/24
1 ZURI Provider 3 BOST Customer 100000 1000 9.0.3.0/24
...
```

How to run your own mini-Internet?

1. Pull from our GitHub page

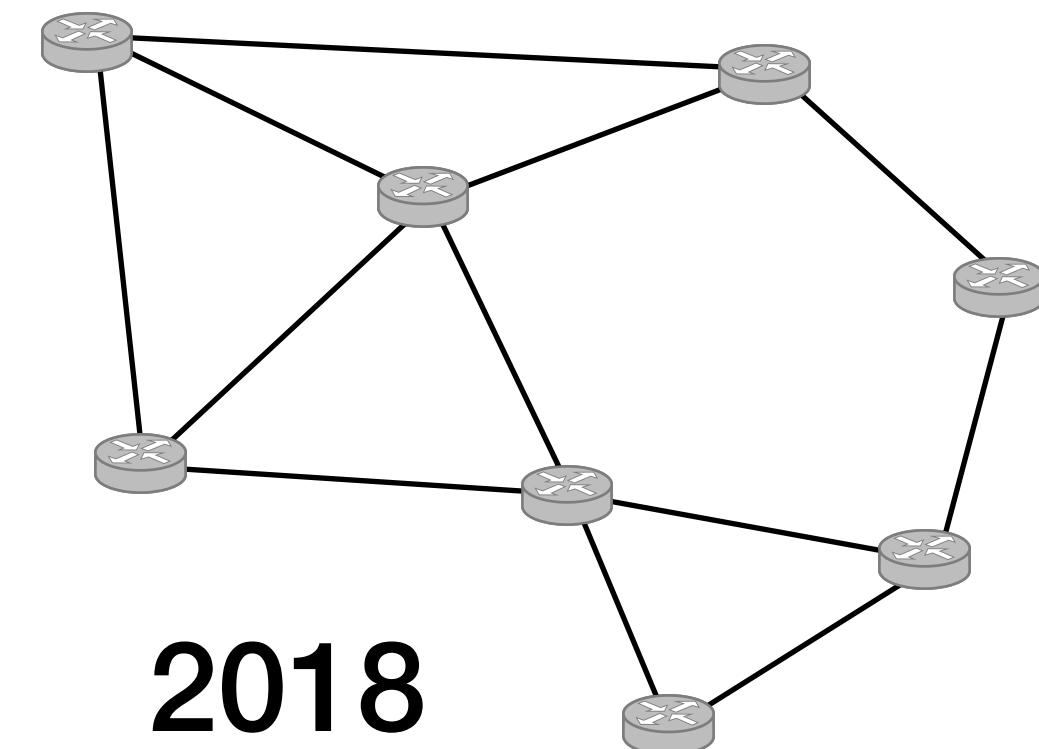
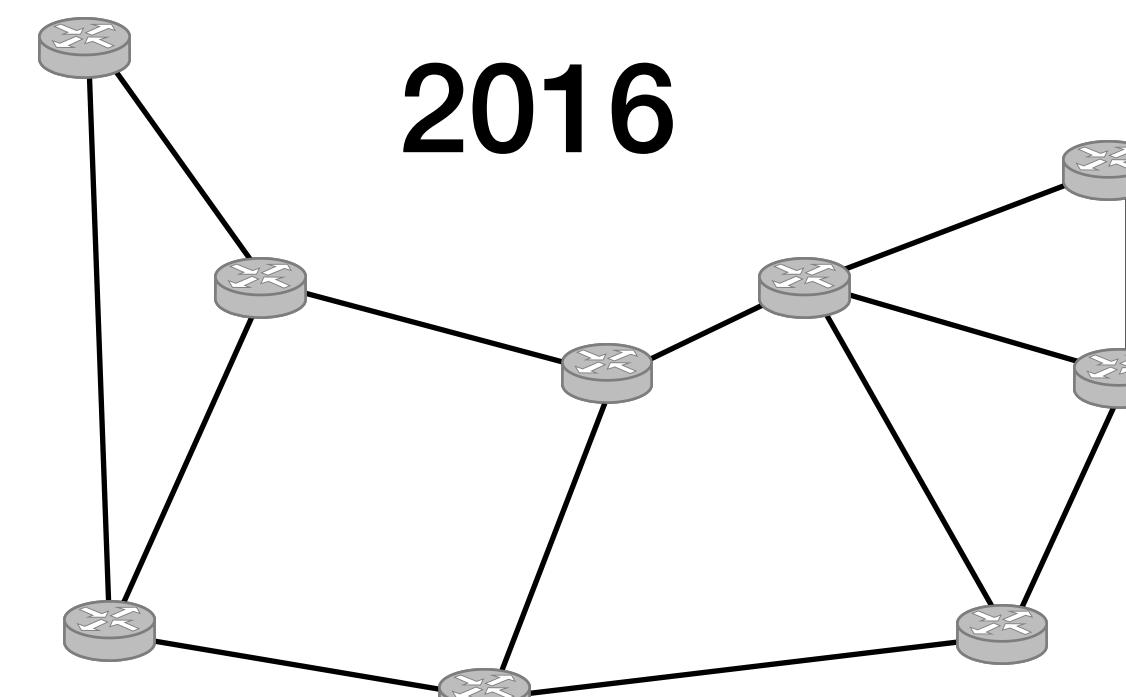
github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

... or use a predefined one:



How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

Execute a single command:

`sudo ./startup.sh`

Wait until your mini-Internet
is completely built
(varies with topology size)

Enjoy!

How to run your own mini-Internet?

1. Pull from our GitHub page

github.com/nsg-ethz/mini_internet_project

2. Follow the documentation

3. Define your topologies

4. Run it on your server

