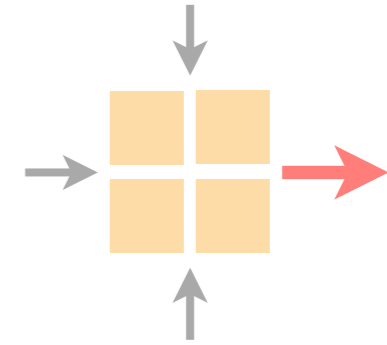


# Communication Networks

Spring 2020



Rüdiger Birkner, TA

Thomas Holterbach, TA

Noah Hütter, TA

Eric Marty, TA

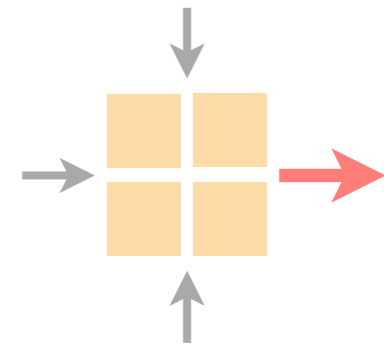
<https://comm-net.ethz.ch/>

ETH Zürich

April 23 2020

# Communication Networks

## Exercise 10



Wrap-up of the routing project

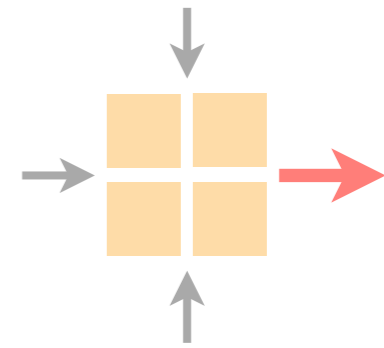
Intro to the reliable transport project

Intro to Python and Git

Current assignment

# Communication Networks

## Exercise 10



**Wrap-up of the routing project**

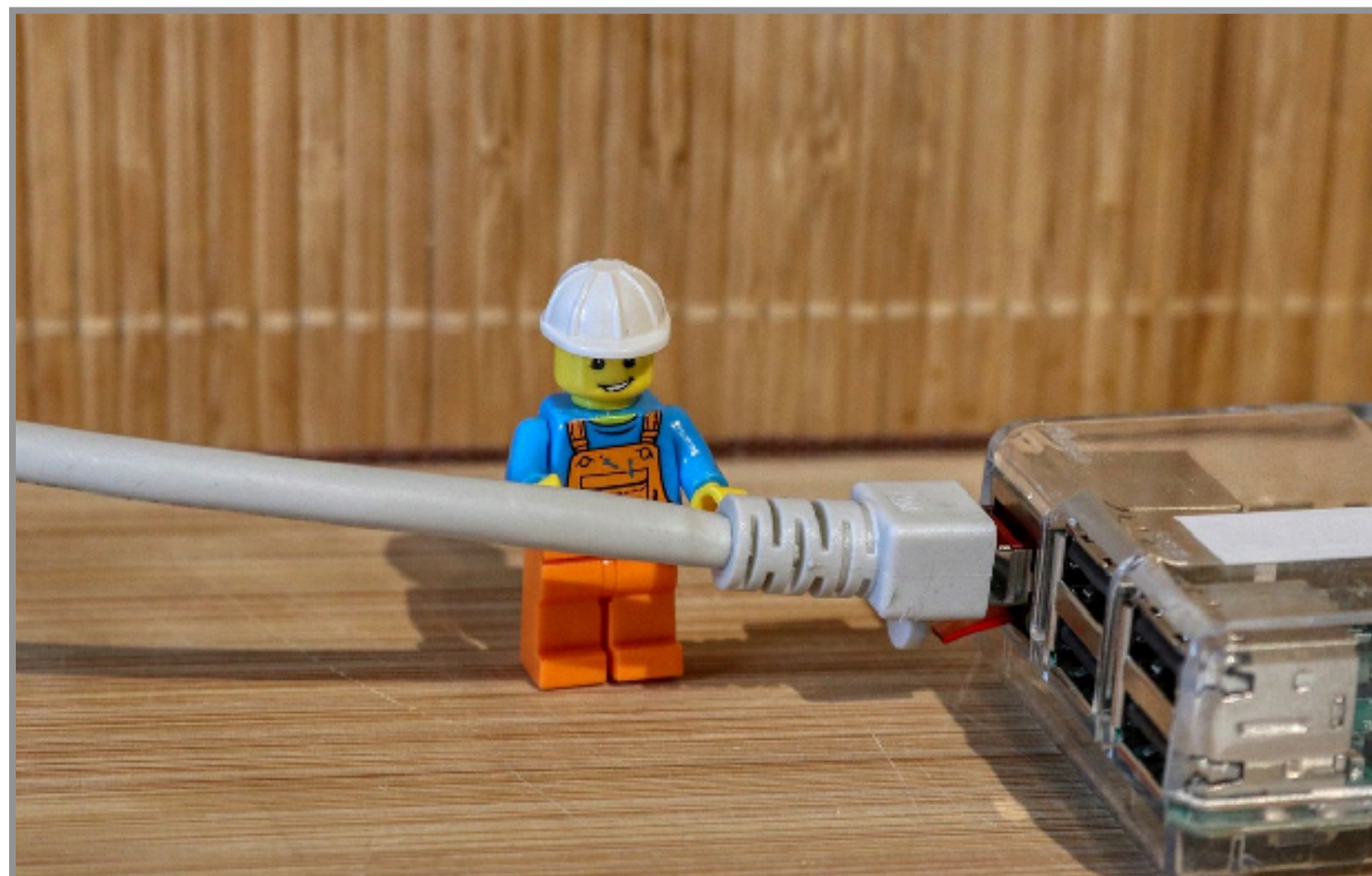
Intro to the reliable transport project

Intro to Python and Git

Current assignment

# Communication Networks 2020

## How we build a mini-Internet



Thomas Holterbach

<https://comm-net.ethz.ch/>

ETH Zurich (D-ITET)

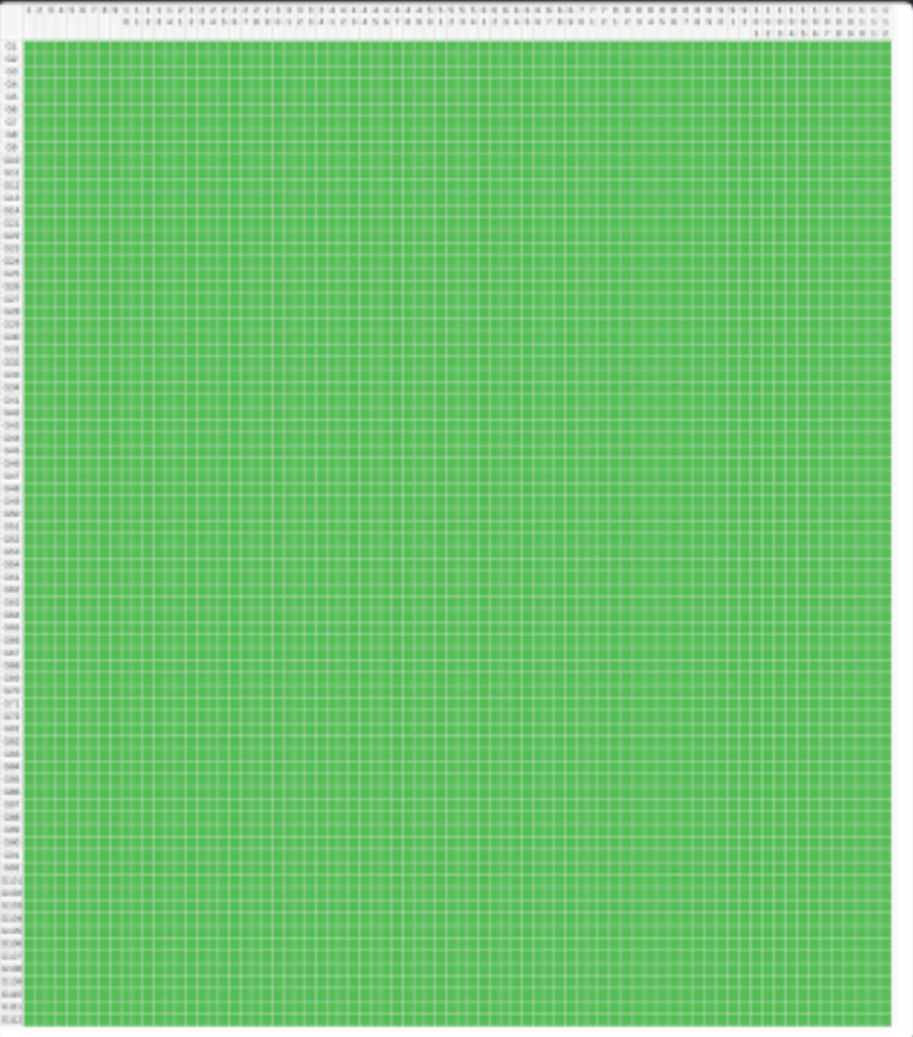
April 23, 2020

# You did it: 100% connectivity!

**Jonas Mehr** Apr 8th at 10:51 AM

We did it!

2020-04-08-105155\_1017x1155\_scrot.png



13 🚀 3 🏆 4 🤖 6 😄 5 📄 28 👤 5 📱 9 📧 2

1 reply

# Also sent to the channel

**Laurent Vanbever** 14 days ago

Very impressive 😄! And, as far as I can remember, in now over 5 iterations of the routing project, this is a first! Good job everyone!

4 😄 18 👤 4 🏆 2 📄 5 📱 1 😄

# Many of you managed to solve the bonus question!

 **Kilian** 11:57 PM  
Hello everyone, there is a surprise GIF at 49.200.30.30 waiting for you

 2  7 


**Congratulations**

[You did it.](#)

You configured VLANs, Gateways, OSPF, BGP and even OpenVPN.

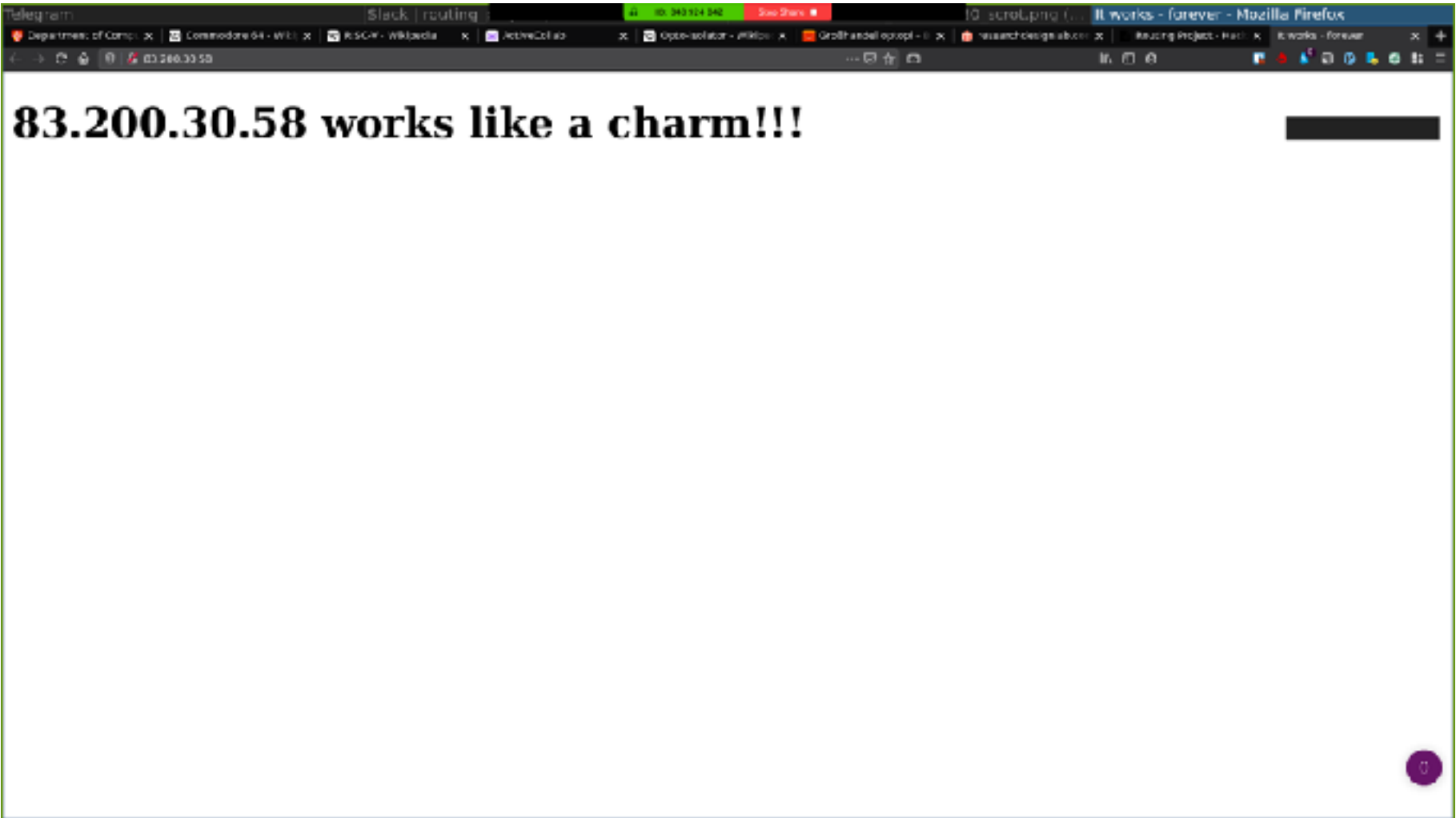
You defended your prefix against some nasty Tier 1 AS hijacking it from you, you dealt with ASes would advertise all their /24 prefixes to you, cluttering your looking glasses and with other ASes that would not stop providing transfer to everyone.

You are 1337.



Here is a party parrot for you, you deserve it!  
(Actually, you deserve a better surprise, but hey, it's bigger than in Slack)

Special Thanks to Prof. Vanbever and his amazing team of TAs, you guys carried us. It was a lot of work, but I really learned a lot in this project. Thank you very much for this amazing opportunity!

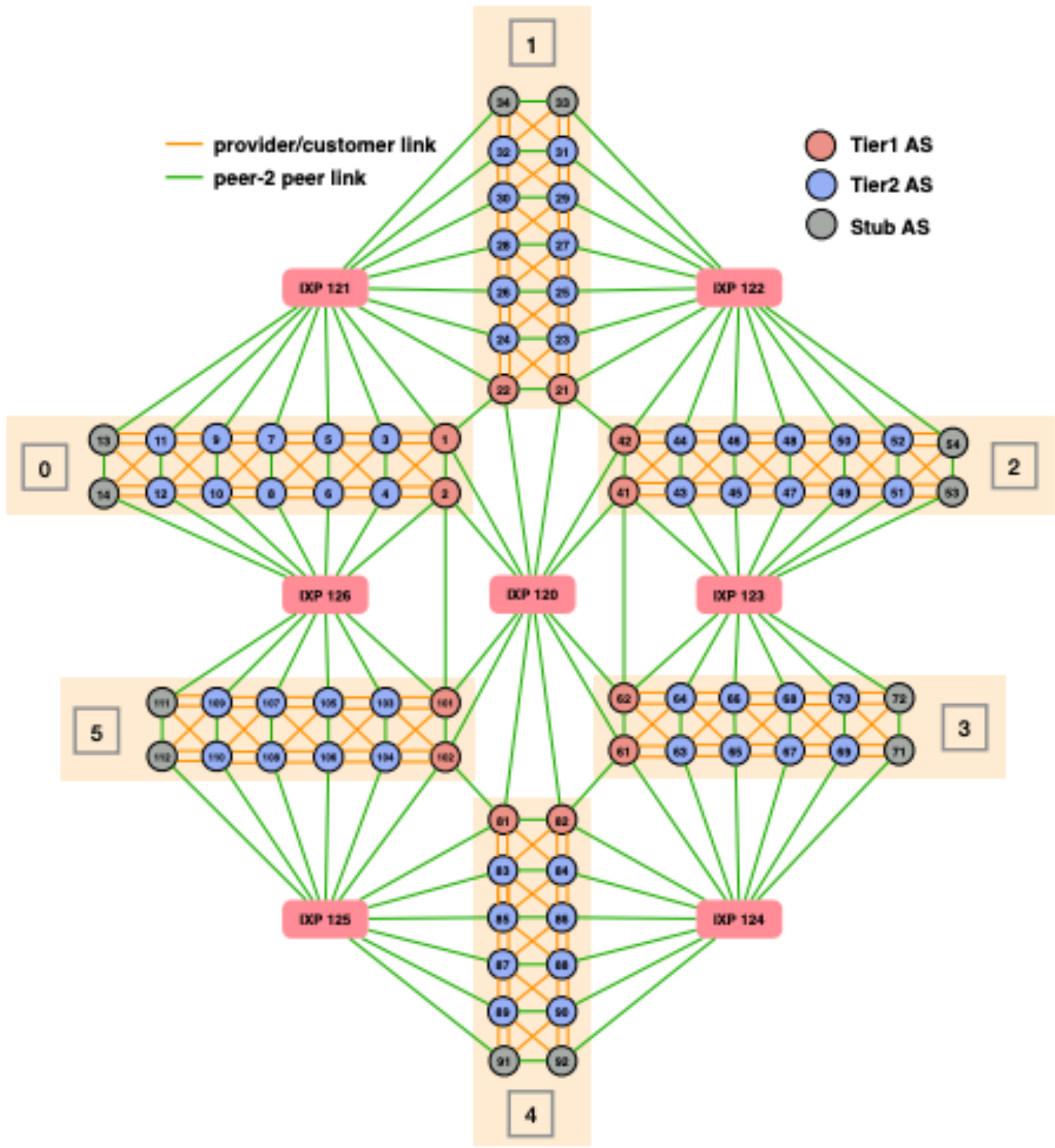


The screenshot shows a terminal window with the following text:

```
83.200.30.58 works like a charm!!!
```

The terminal window also shows a list of open tabs at the top, including 'Slack | routing', '83.200.30.58', 'id -scrot.png', and 'it works - forever - Mozilla Firefox'. The terminal output is a single line of text indicating a successful connection to the IP address 83.200.30.58.

# Today, we will see how we designed the mini-Internet



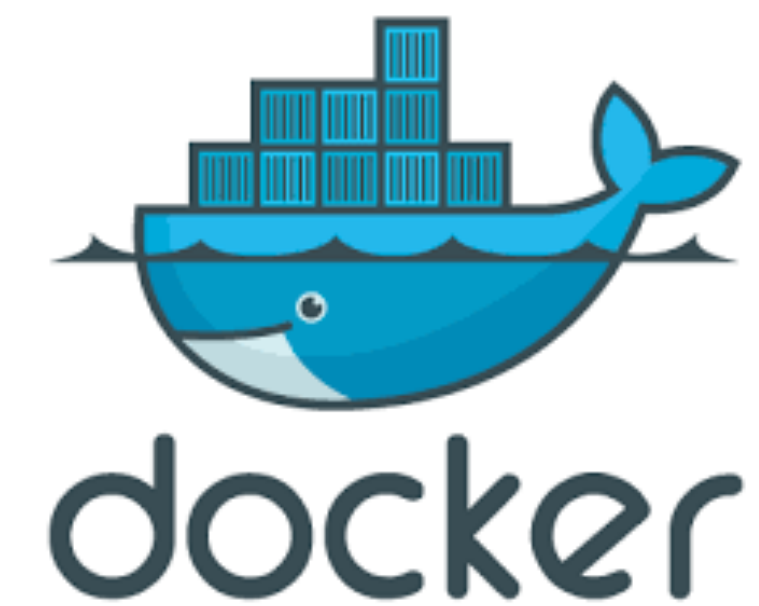
snowball.ethz.ch

# We rely on virtualisation

Option #1: virtual machines



Option #2: linux containers



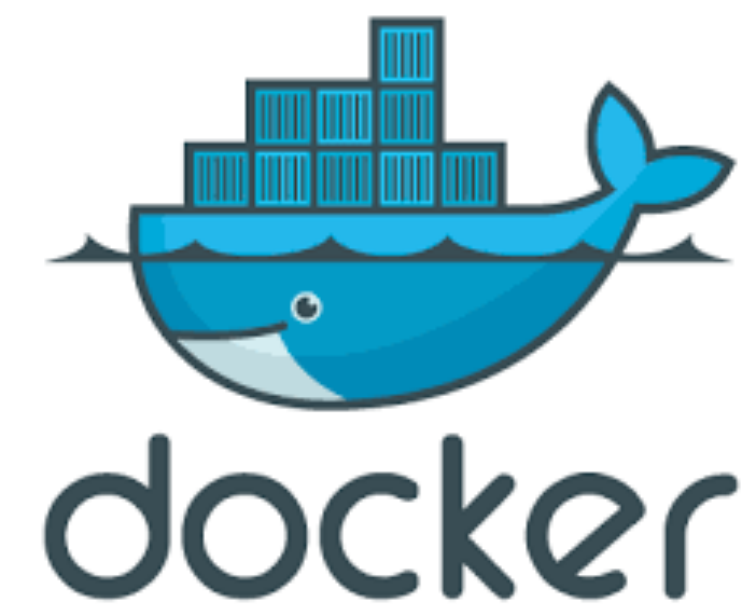


# We rely on virtualisation

## Option #1: virtual machines



## Option #2: linux containers



we used VMs between  
2016 and 2019

# We rely on virtualisation

## Option #1: virtual machines



we used VMs between  
2016 and 2019

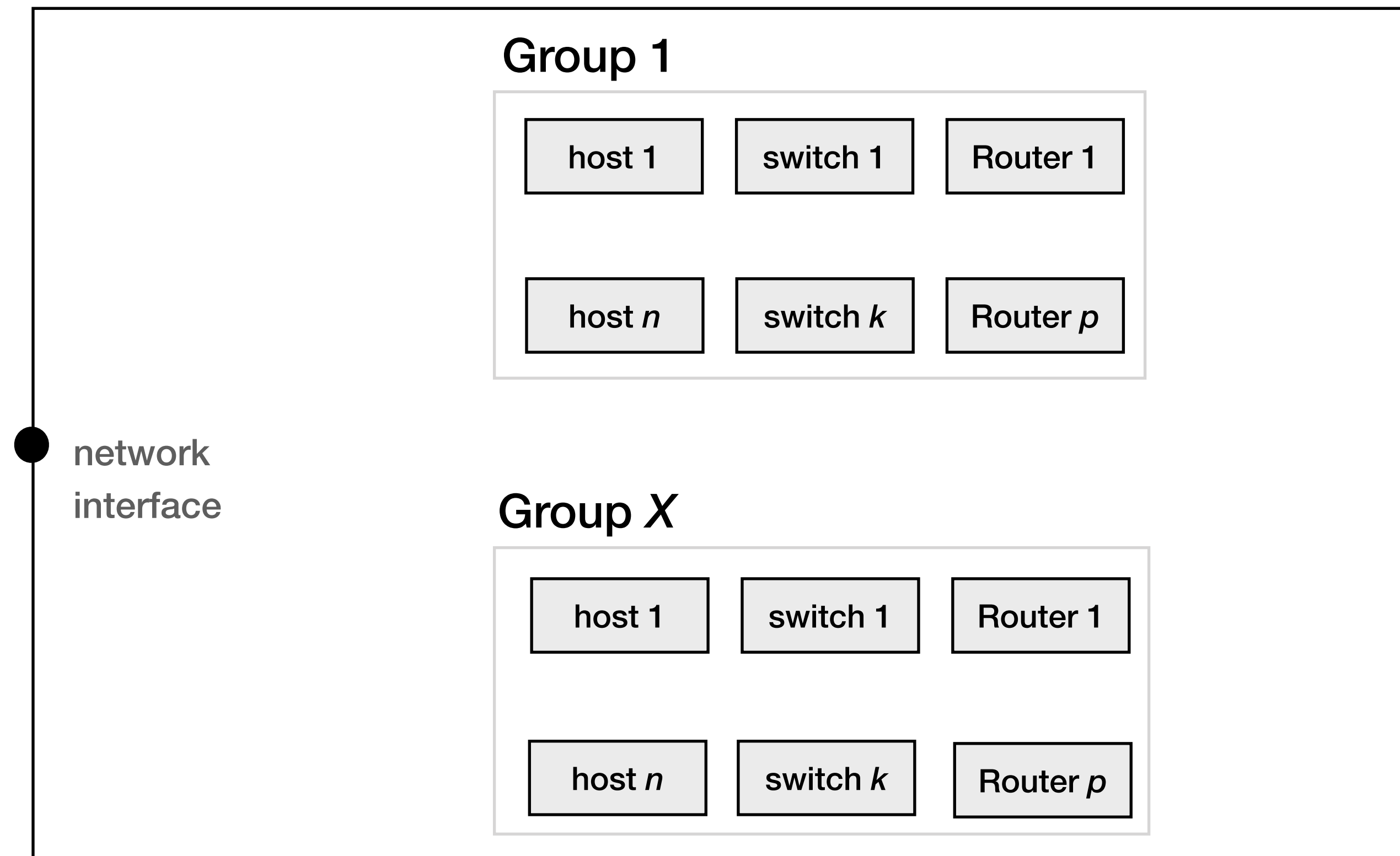
## Option #2: linux containers



This year

Each router, switch and host runs in its dedicated container

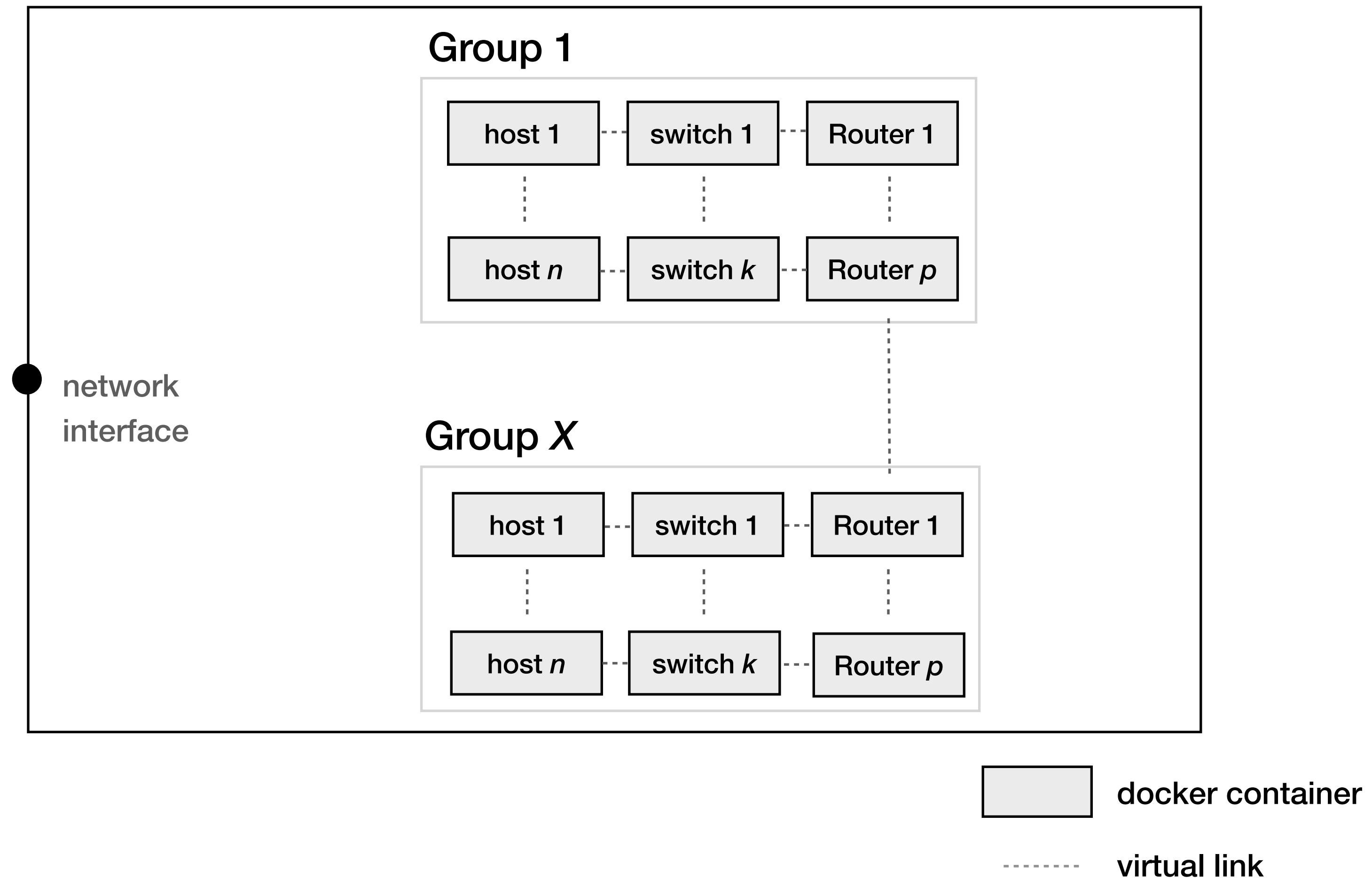
snowball.ethz.ch



 docker container

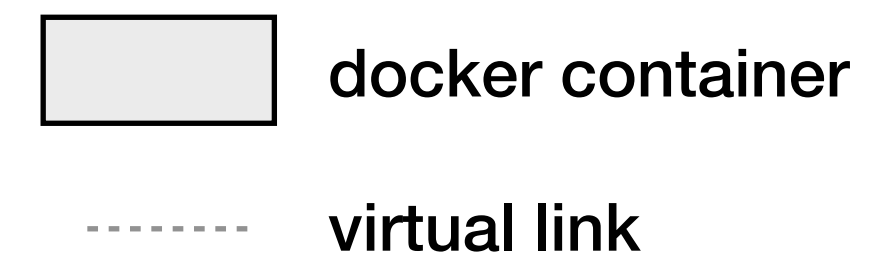
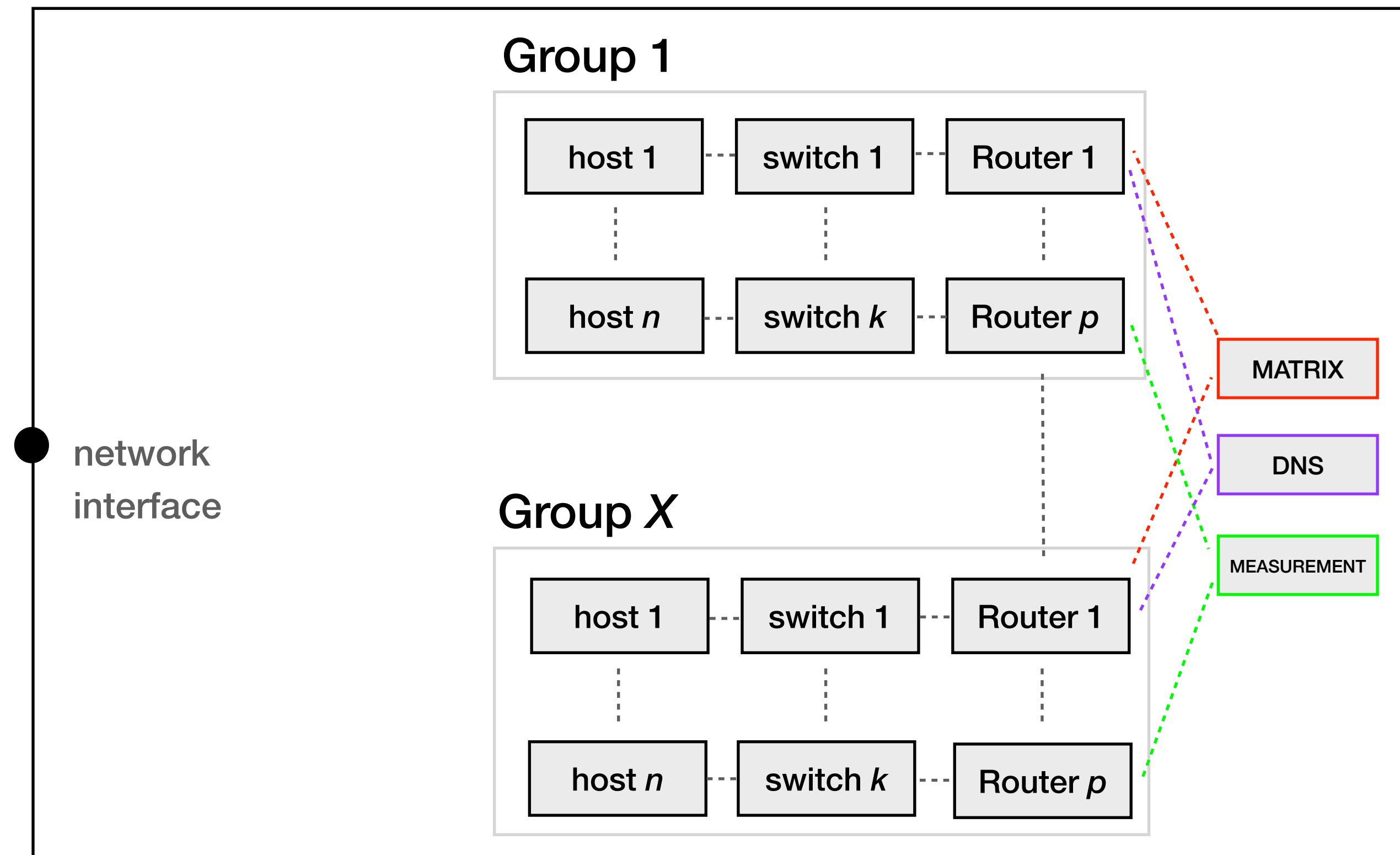
Each router, switch and host runs in its dedicated container  
We virtually connect the containers to build the mini-Internet

snowball.ethz.ch



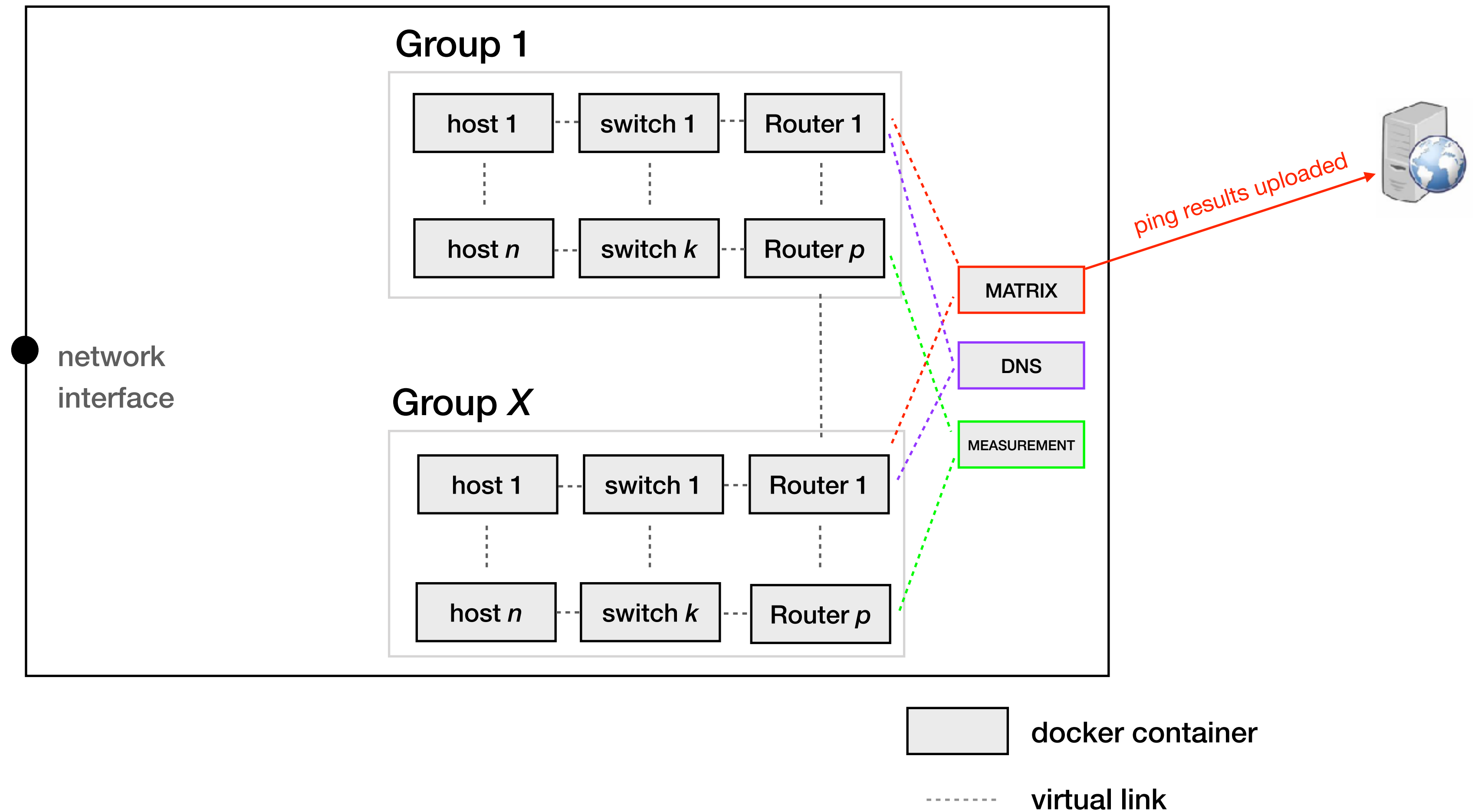
We use additional containers for the different monitoring services

snowball.ethz.ch



# We use additional containers for the different monitoring services

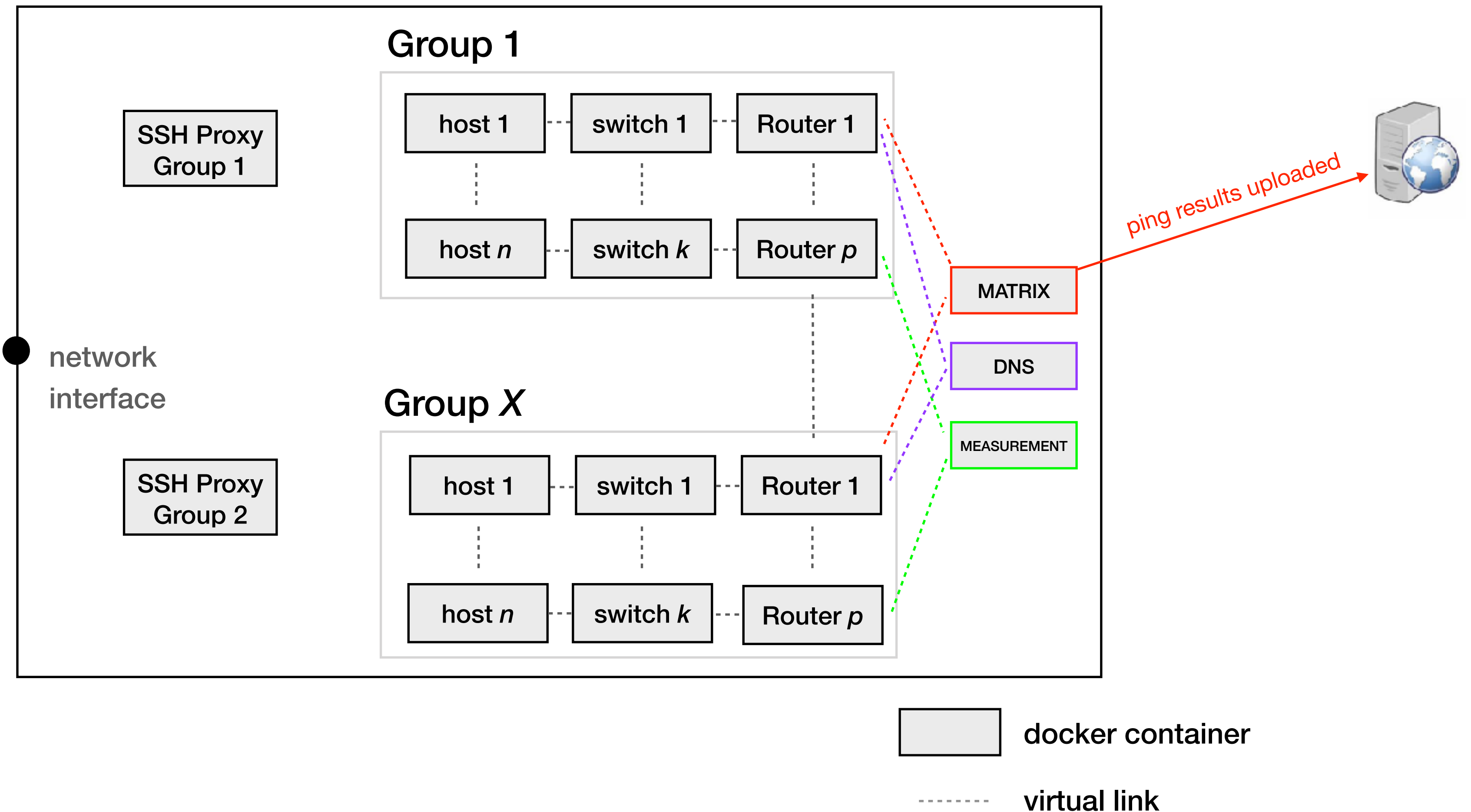
snowball.ethz.ch



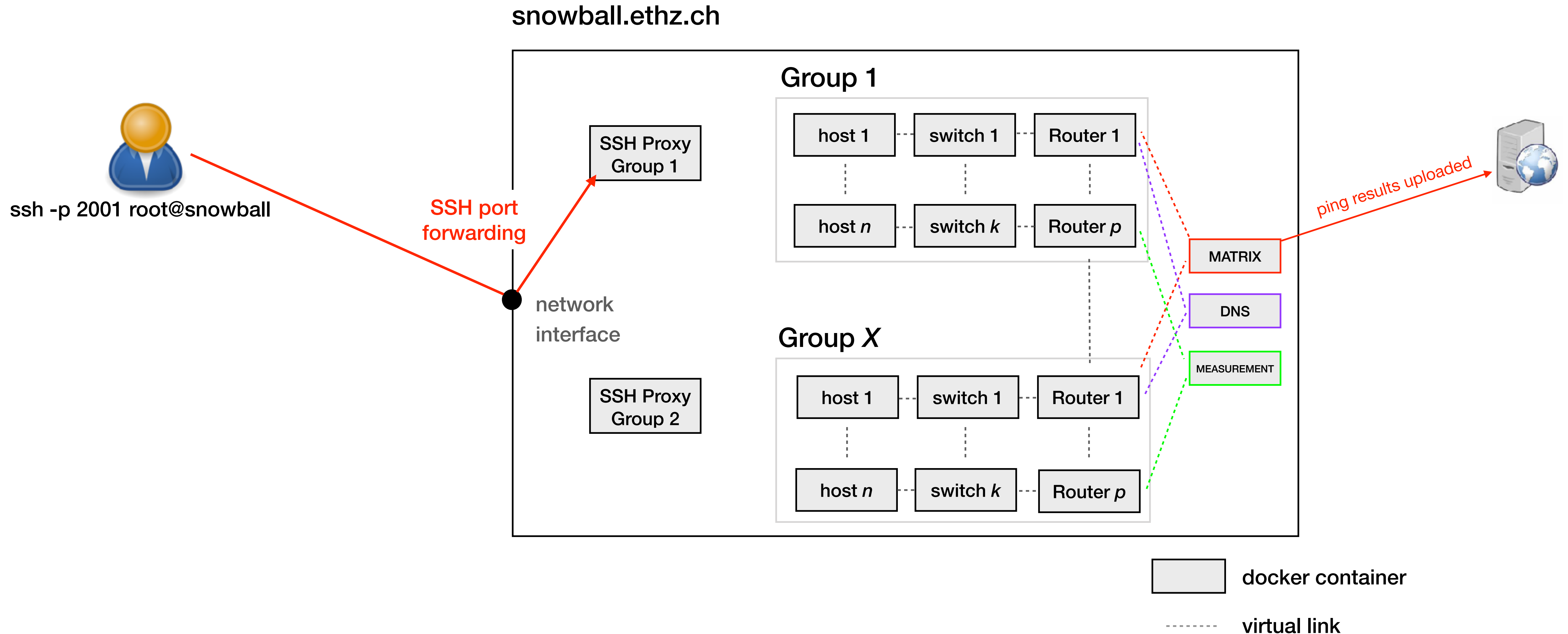
We use "proxy" containers so that you can only access *your* virtual devices

  
`ssh -p 2001 root@snowball`

snowball.ethz.ch

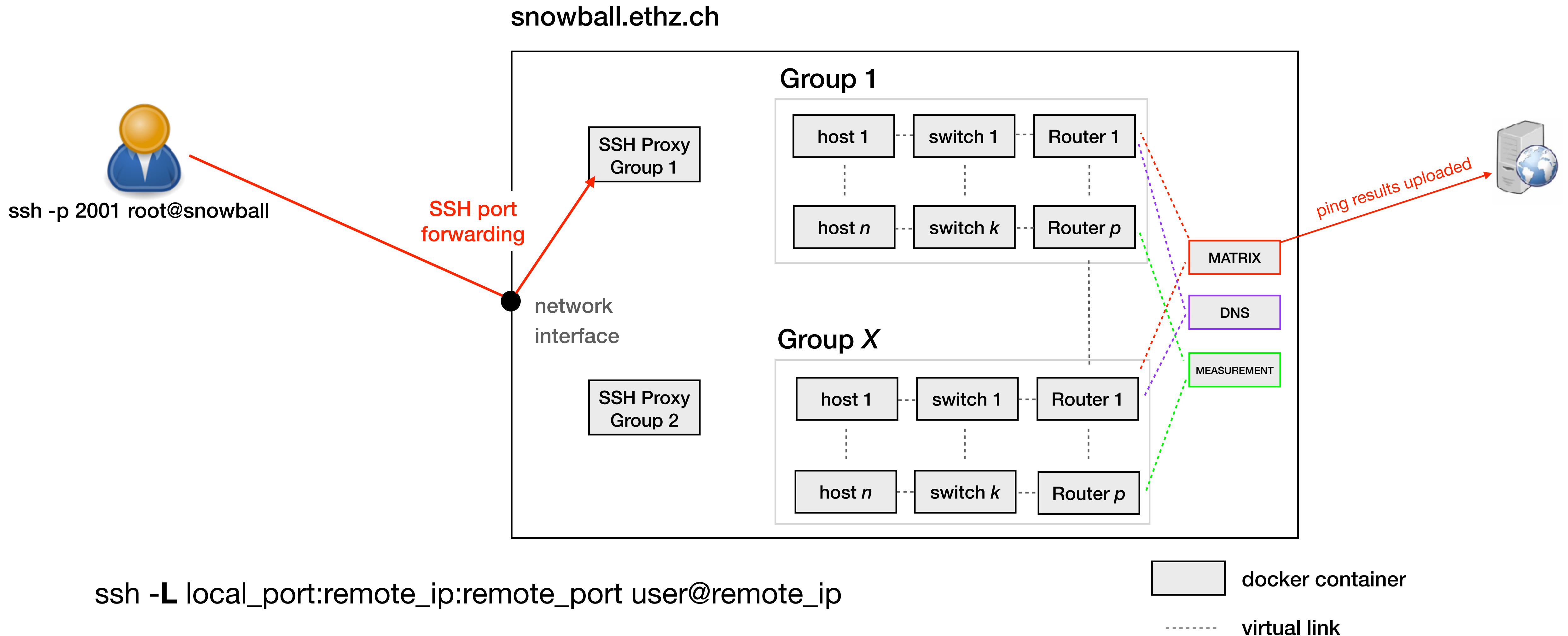


We use "proxy" containers so that you can only access *your* virtual devices



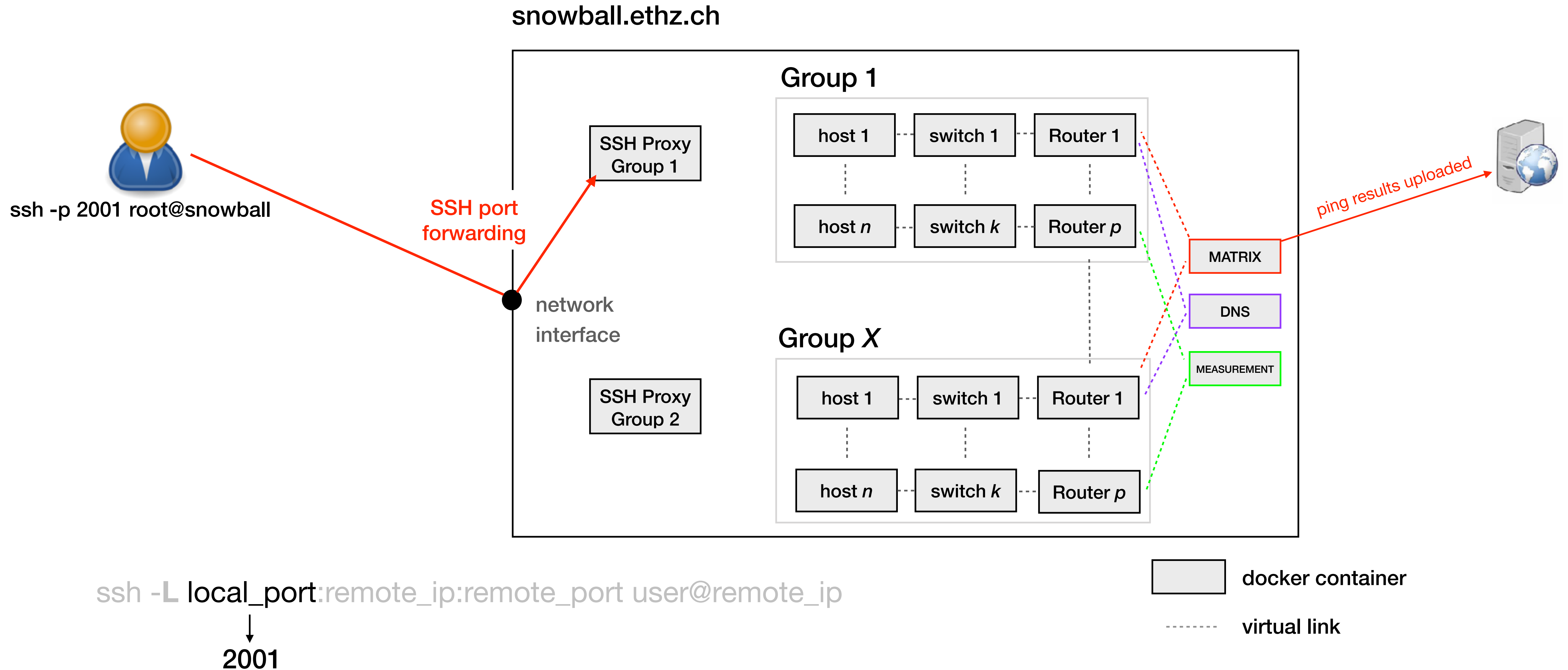


We use "proxy" containers so that you can only access *your* virtual devices

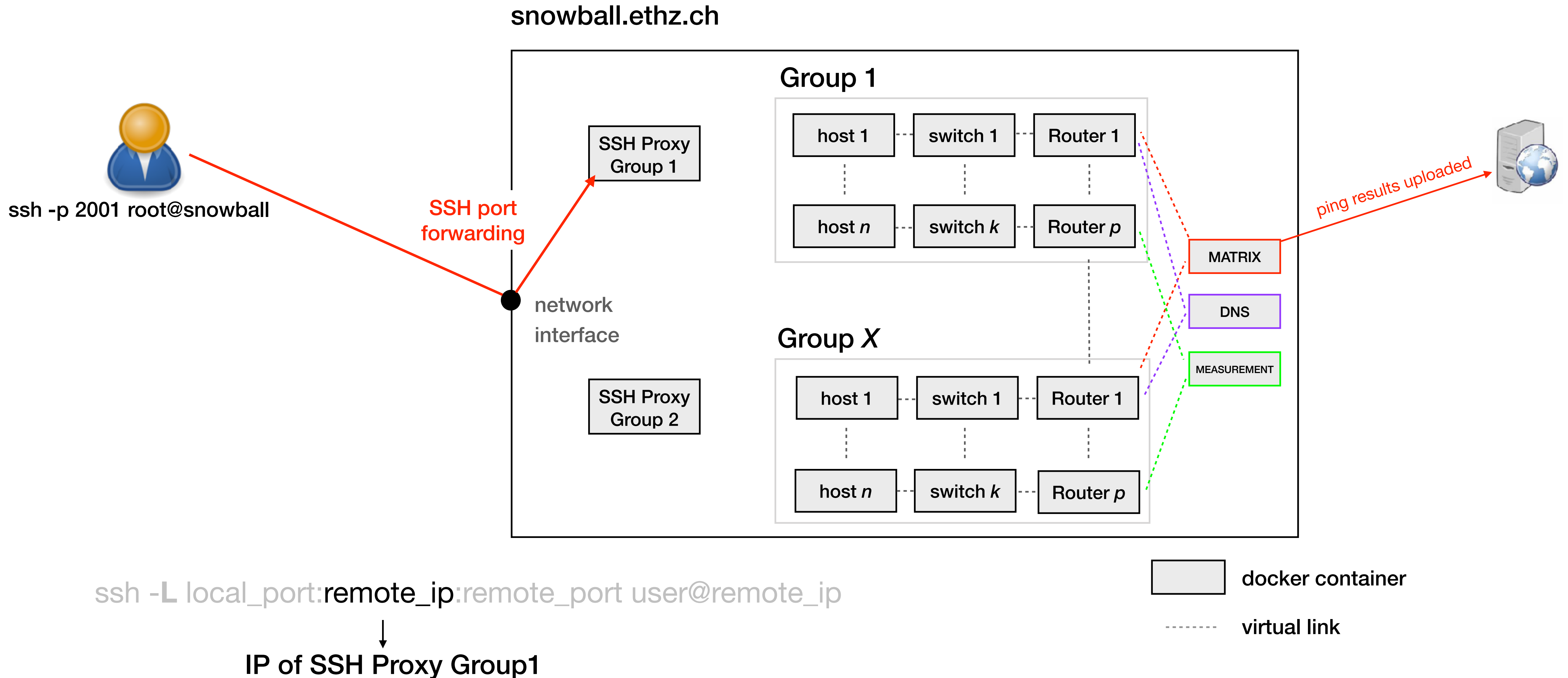


`ssh -L local_port:remote_ip:remote_port user@remote_ip`

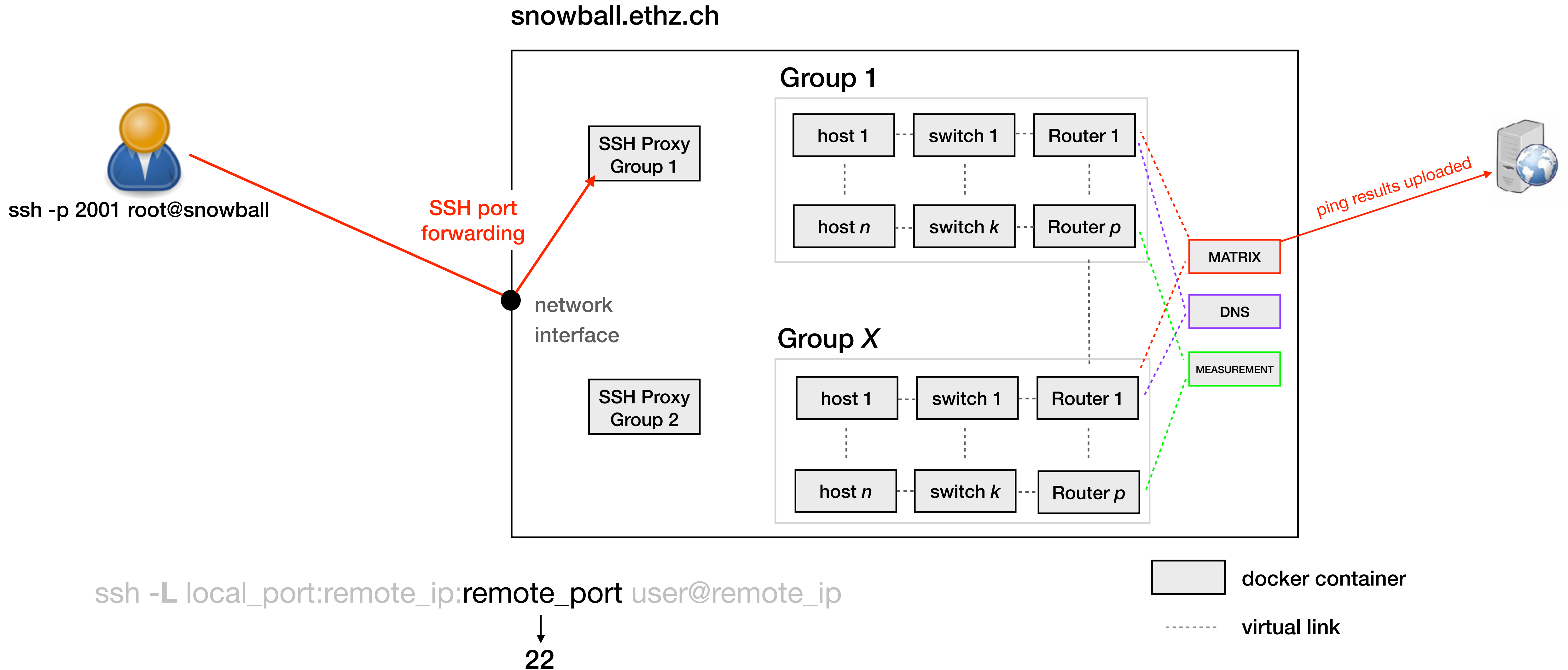
We use "proxy" containers so that you can only access *your* virtual devices



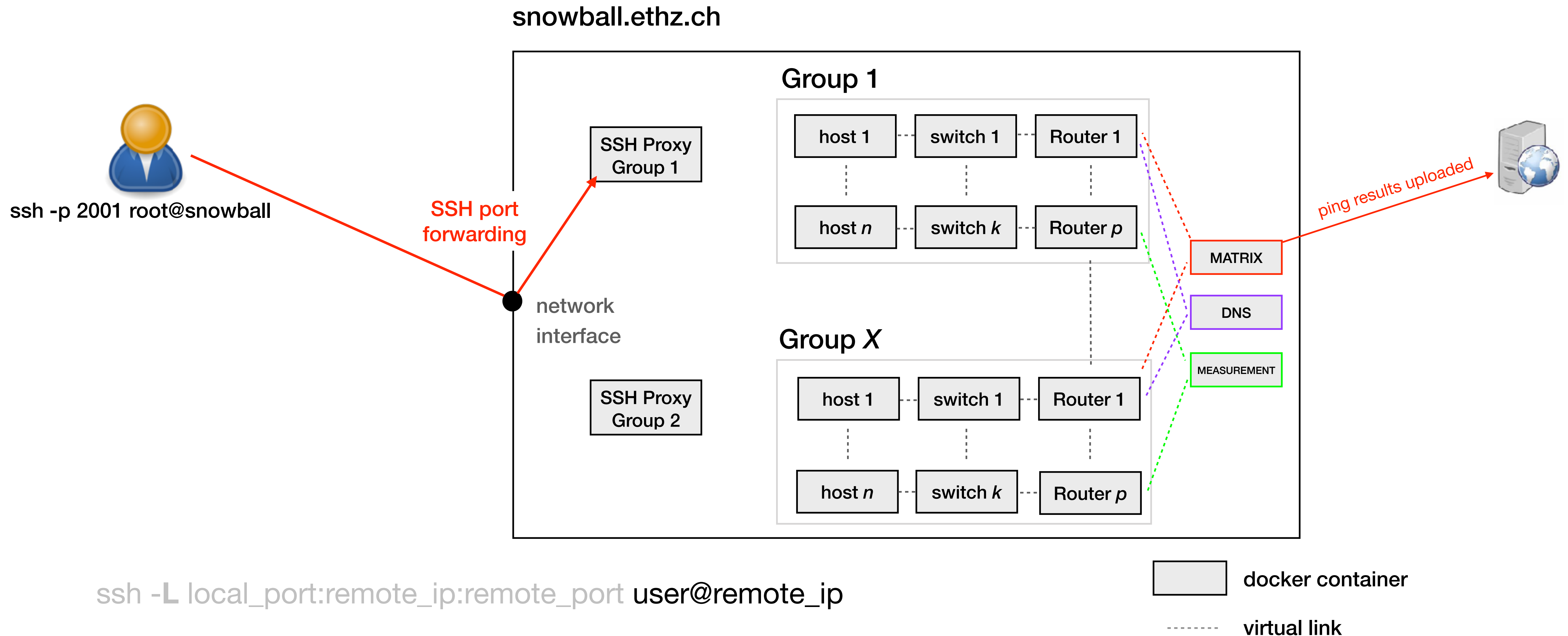
We use "proxy" containers so that you can only access *your* virtual devices



We use "proxy" containers so that you can only access *your* virtual devices

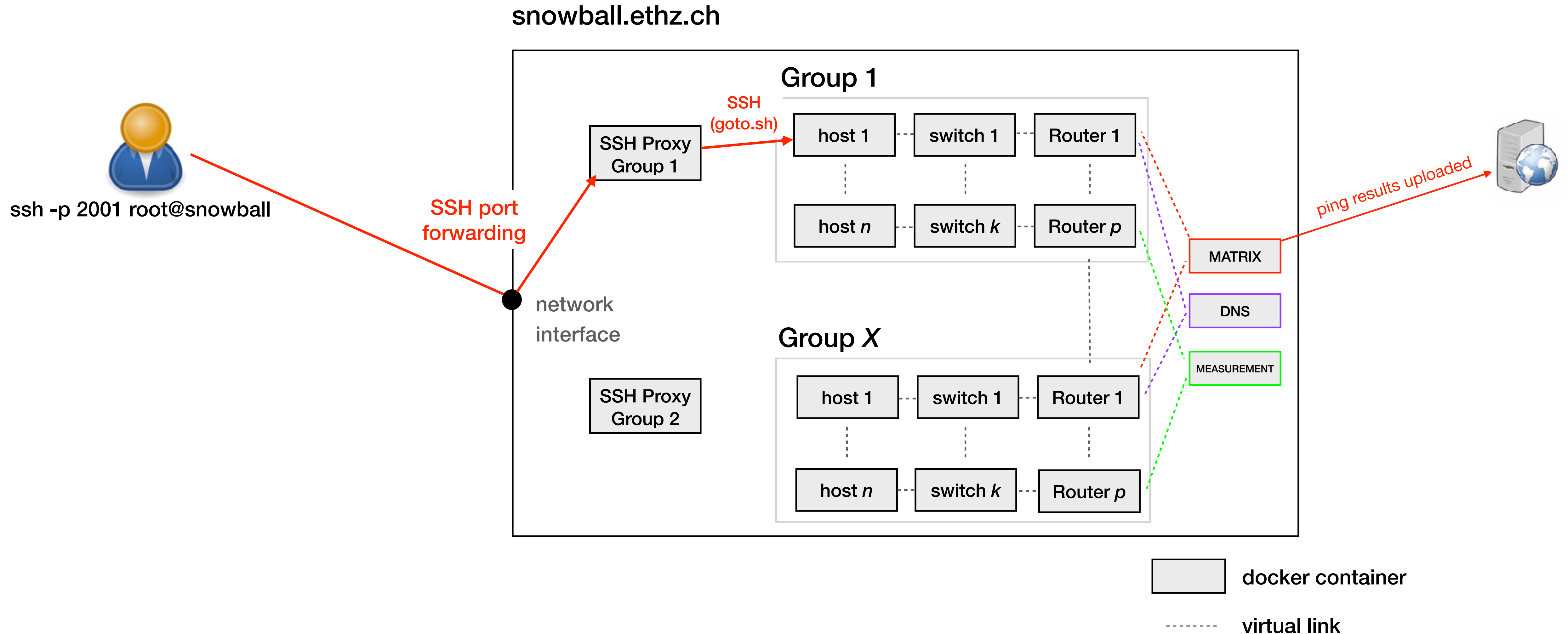


We use "proxy" containers so that you can only access *your* virtual devices

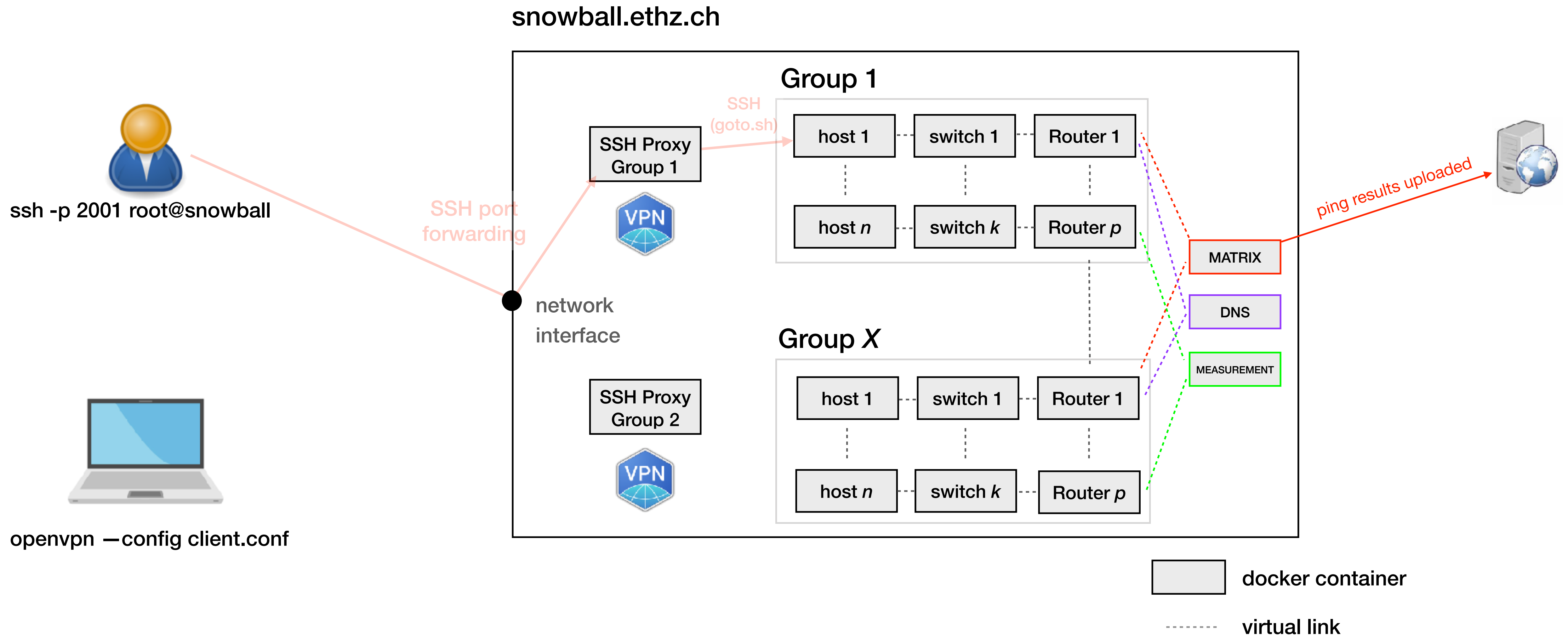


`ssh -L local_port:remote_ip:remote_port user@remote_ip`

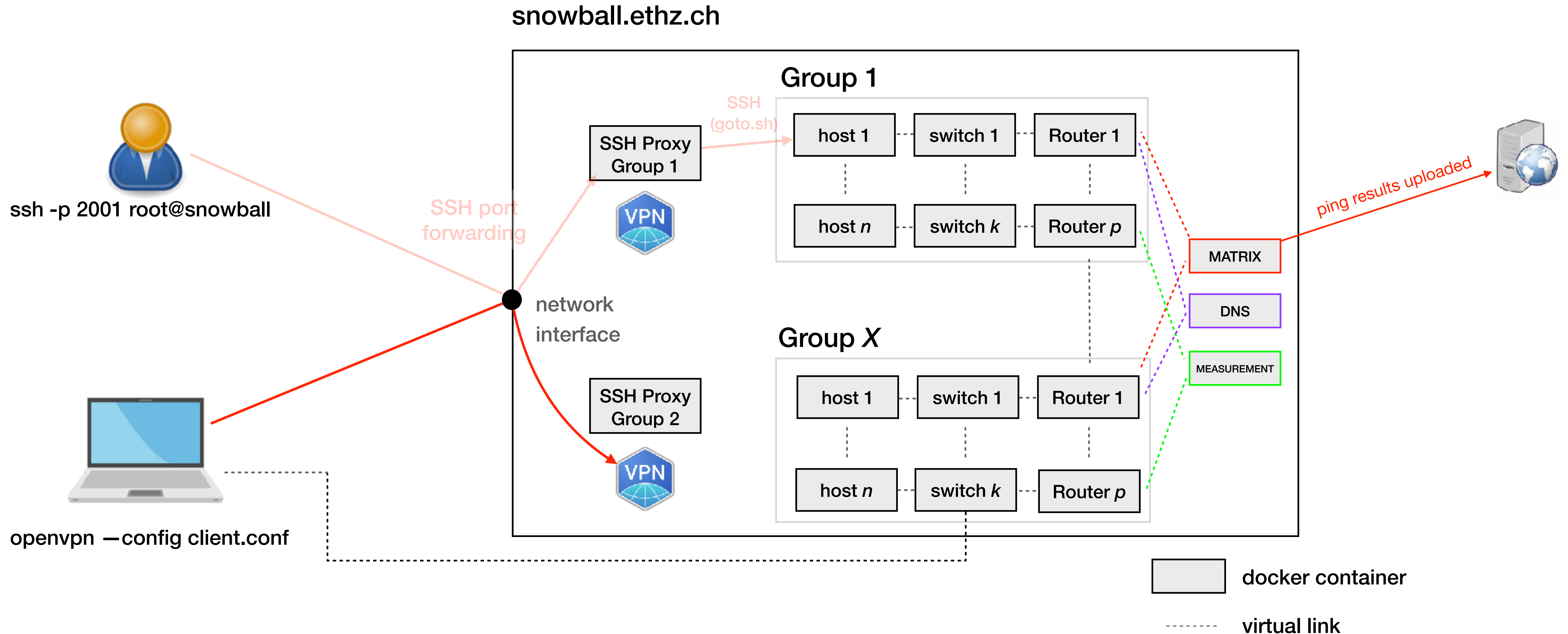
We use "proxy" containers so that you can only access *your* virtual devices



We use "proxy" containers so that you can only access *your* virtual devices



We use "proxy" containers so that you can only access *your* virtual devices





# Our server can easily run a 78-ASes mini-Internet

```
 1 [|||||] 30.3% 13 [|||||] 21.2% 25 [|||||] 37.0% 37 [|||||] 25.9%
 2 [|||||] 29.3% 14 [|||||] 17.0% 26 [|||||] 19.4% 38 [|||||] 24.2%
 3 [|||||] 35.4% 15 [|||||] 27.5% 27 [|||||] 40.0% 39 [|||||] 22.6%
 4 [|||||] 31.0% 16 [|||||] 24.9% 28 [|||||] 22.3% 40 [|||||] 15.6%
 5 [|||||] 28.9% 17 [|||||] 32.4% 29 [|||||] 32.4% 41 [|||||] 23.5%
 6 [|||||] 30.3% 18 [|||||] 22.5% 30 [|||||] 16.2% 42 [|||||] 16.2%
 7 [|||||] 20.7% 19 [|||||] 20.6% 31 [|||||] 32.6% 43 [|||||] 15.0%
 8 [|||||] 24.5% 20 [|||||] 34.9% 32 [|||||] 38.0% 44 [|||||] 12.2%
 9 [|||||] 24.3% 21 [|||||] 38.1% 33 [|||||] 27.9% 45 [|||||] 19.0%
10 [|||||] 47.4% 22 [|||||] 32.4% 34 [|||||] 20.0% 46 [|||||] 33.3%
11 [|||||] 30.4% 23 [|||||] 64.0% 35 [|||||] 37.6% 47 [|||||] 24.4%
12 [|||||] 24.1% 24 [|||||] 48.6% 36 [|||||] 23.5% 48 [|||||] 27.4%
Mem[|||||] 51.2G/252G Tasks: 10137, 28281 thr; 7 running
Swp[ ] 0K/29.8G Load average: 25.97 28.57 28.71
Uptime: 60 days, 08:18:51
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
22250	root	10	-10	4287M	1211M	9356	S	1.3	0.5	10h29:19	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153400	root	10	-10	4287M	1211M	9356	S	1.3	0.5	1h07:32	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153402	root	10	-10	4287M	1211M	9356	S	0.8	0.5	55:15.15	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153403	root	10	-10	4287M	1211M	9356	S	0.8	0.5	45:08.96	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153404	root	10	-10	4287M	1211M	9356	S	0.8	0.5	37:27.11	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153405	root	10	-10	4287M	1211M	9356	S	0.4	0.5	31:57.49	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153410	root	10	-10	4287M	1211M	9356	S	0.4	0.5	27:29.55	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153411	root	10	-10	4287M	1211M	9356	S	0.4	0.5	24:14.55	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153413	root	10	-10	4287M	1211M	9356	S	0.4	0.5	21:29.24	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153418	root	10	-10	4287M	1211M	9356	S	0.0	0.5	19:16.55	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153419	root	10	-10	4287M	1211M	9356	S	0.8	0.5	17:37.17	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153420	root	10	-10	4287M	1211M	9356	S	0.4	0.5	16:10.48	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153425	root	10	-10	4287M	1211M	9356	S	0.0	0.5	15:04.09	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153426	root	10	-10	4287M	1211M	9356	S	0.4	0.5	14:13.02	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153427	root	10	-10	4287M	1211M	9356	S	0.0	0.5	13:24.57	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153428	root	10	-10	4287M	1211M	9356	S	0.4	0.5	12:43.77	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153429	root	10	-10	4287M	1211M	9356	S	0.4	0.5	12:09.53	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153433	root	10	-10	4287M	1211M	9356	S	0.4	0.5	11:42.22	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153434	root	10	-10	4287M	1211M	9356	S	0.0	0.5	11:18.37	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153438	root	10	-10	4287M	1211M	9356	S	0.0	0.5	10:56.20	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153441	root	10	-10	4287M	1211M	9356	S	0.4	0.5	10:34.89	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153444	root	10	-10	4287M	1211M	9356	S	0.0	0.5	10:19.60	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153445	root	10	-10	4287M	1211M	9356	S	0.0	0.5	10:00.50	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153446	root	10	-10	4287M	1211M	9356	S	0.0	0.5	9:44.31	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f
153448	root	10	-10	4287M	1211M	9356	S	0.0	0.5	9:31.36	ovs-vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-f

For further information about the mini-Internet: [mini-inter.net](https://mini-inter.net)

Open source implementation  
~3700 lines of bash

nsg-ethz / mini\_internet\_project

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

The official repository of the mini-Internet exercise.

273 commits 2 branches 0 packages 0 releases 3 contributors GPL-3.0

File	Description	Age
2019_assignment_eth	type assignment folder	27 days ago
2020_assignment_eth	Create README.md	last month
platform	do not run the matrix script automatically	5 days ago
.gitignore	added .gitignore file	6 months ago
LICENSE	Create LICENSE	5 months ago
README.md	Update README.md	7 days ago

### 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.

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 [2020\\_assignment\\_eth](#), we further describe how we used the mini-Internet at ETH in the 2020 iteration of our Communication Networks lecture. While the mini-Internet project works well for our introductory class, observe that it can be adapted for various teaching objectives.

#### Build your mini-Internet

With this platform, you can easily build your own mini-Internet, tailored for your teaching objectives. The documentation as well as the source code of the platform can be found in the [platform](#) directory. In a nutshell, after defining your topology in configuration files, you

# For further information about the mini-Internet: [mini-inter.net](https://mini-inter.net)

Open source implementation  
~3700 lines of bash

Published in SIGCOMM CCR'20

Youtube video

**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 component 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.

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 [2020\\_assignment\\_eth](#), we further describe how we used the mini-Internet at ETH in the 2020 iteration of our Communication Networks lecture. While the mini-Internet project works well for our introductory class, observe that it can be adapted for various teaching objectives.

**Build your mini-Internet**

With this platform, you can easily build your own mini-Internet, tailored for your teaching objectives. The documentation as well as the source code of the platform can be found in the [platform](#) directory. In a nutshell, after defining your topology in configuration files, you

## An Open Platform to Teach How the Internet Practically Works

[mini-inter.net](https://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

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.

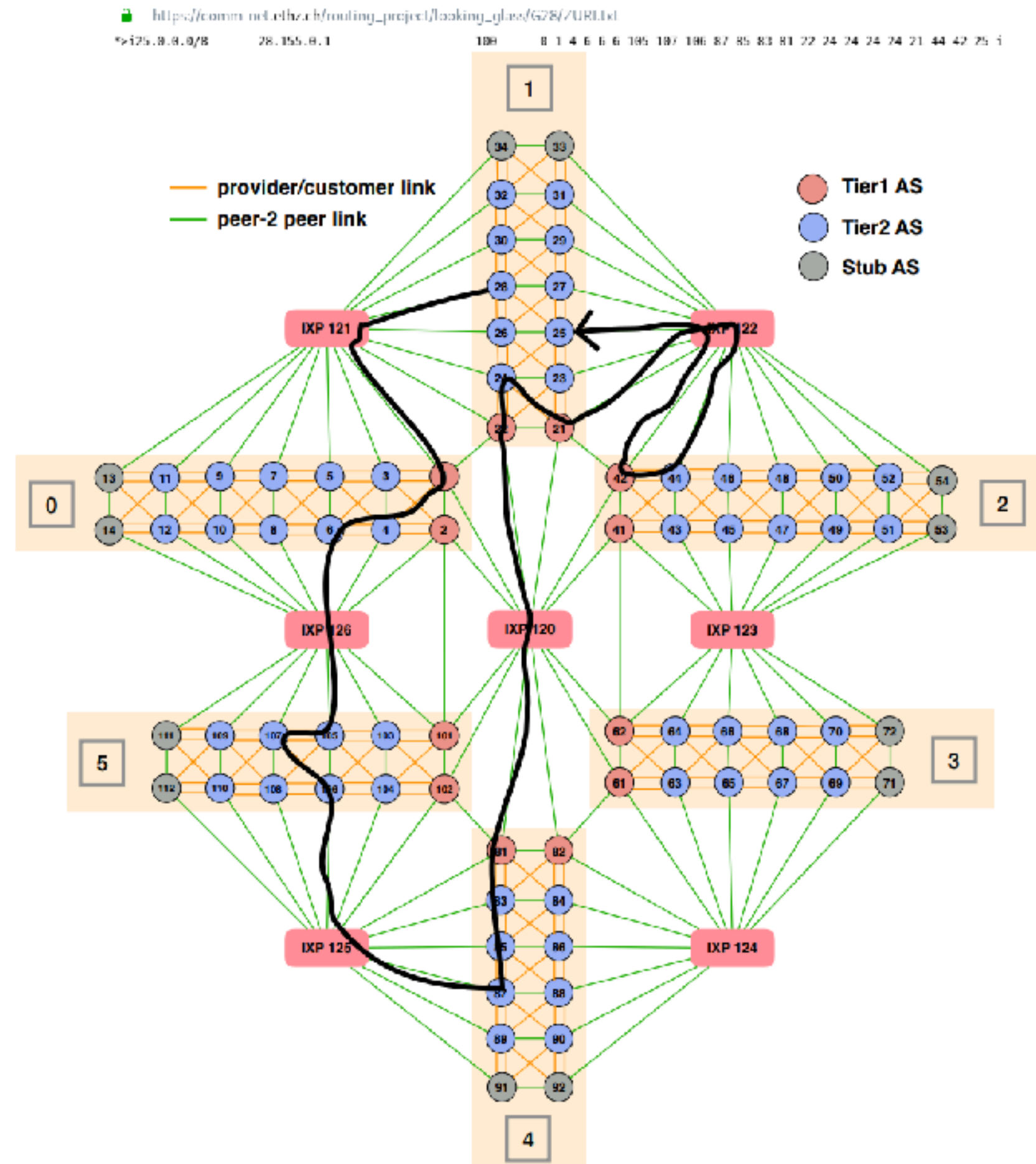
First, we aimed at faithfully emulating the real Internet infrastructure. To do so, we rely on (open-source) switching and routing software implementing the most well-known protocols (e.g., STP, OSPF, BGP). We also rely on virtualization (containers) to interconnect *many* instances (100+) of these software. 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



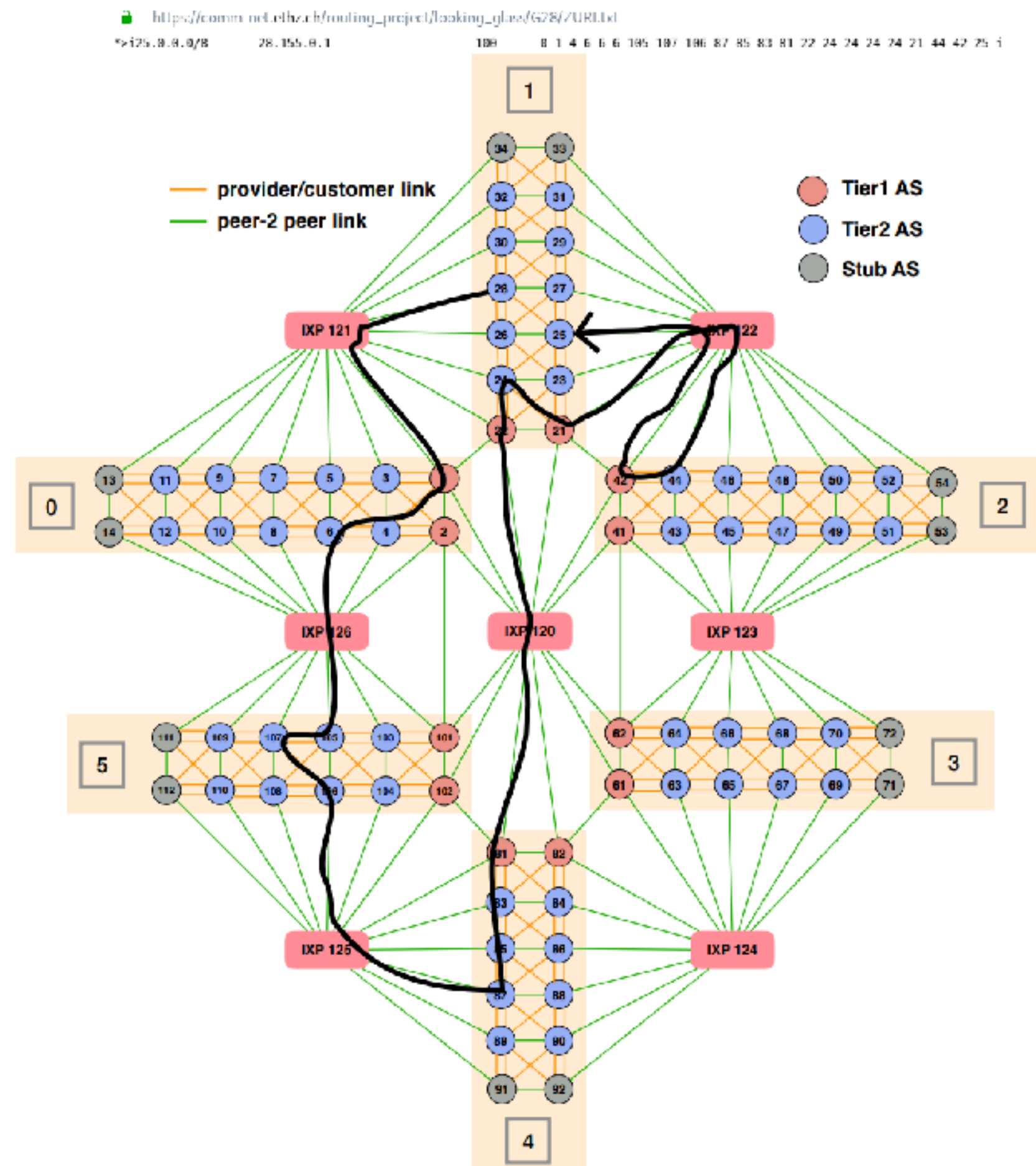
Every year we improve the project

# Every year we improve the project



We still too often observe such incorrect paths

# Every year we improve the project



We still too often observe such incorrect paths

We are designing a visualisation framework to help students detecting those incorrect paths

Every year we improve the project

Although you eliminated the hijacker very well...



Every year we improve the project

Although you eliminated the hijacker very well...

...It is always better to prevent a hijack before it actually happens





Every year we improve the project

Although you eliminated the hijacker very well...



...It is always better to prevent a hijack before it actually happens

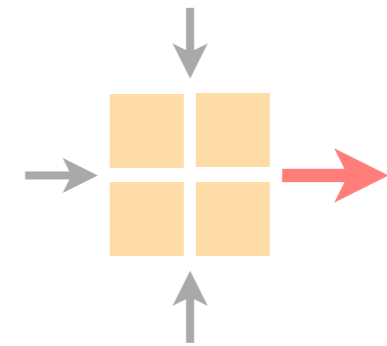
We plan to implement the RPKI infrastructure into the mini-internet so that you can validate the origin of the BGP routes

We offer this as a semester thesis, check out our website for further information!

Please let us know if you have any feedback  
or ideas on how to improve the project :-)

# Communication Networks

## Exercise 10



Wrap-up of the routing project

**Intro to the reliable transport project**

Intro to Python and Git

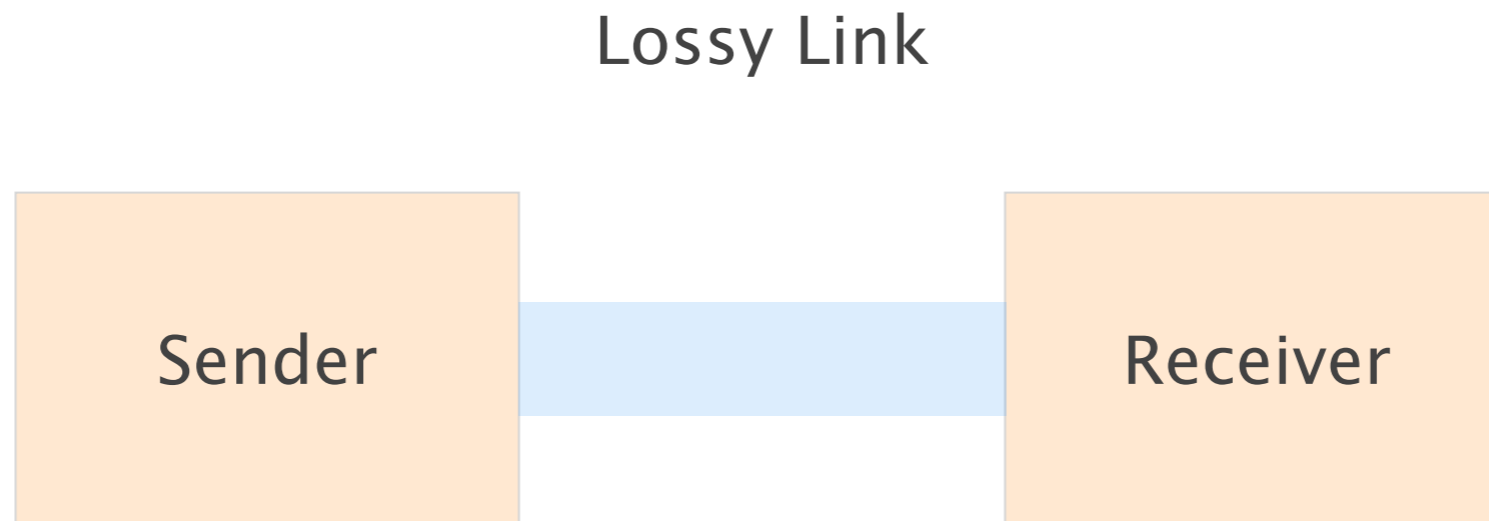
Current assignment

Implement your own Reliable Transport Protocol

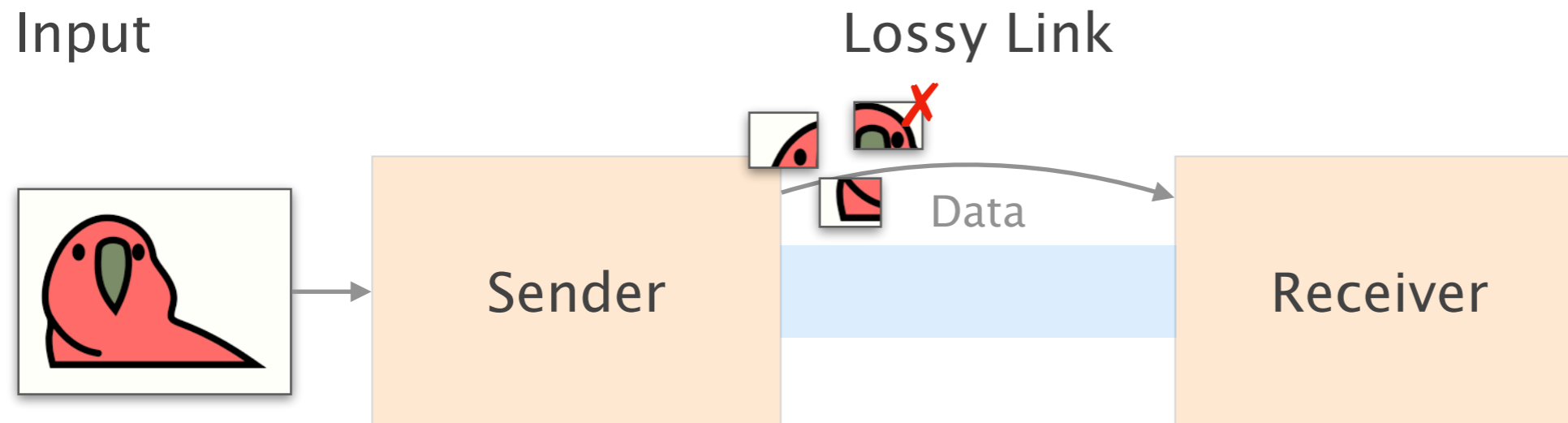
# Implement your own **Reliable** Transport Protocol

recover from packet loss  
and reordering

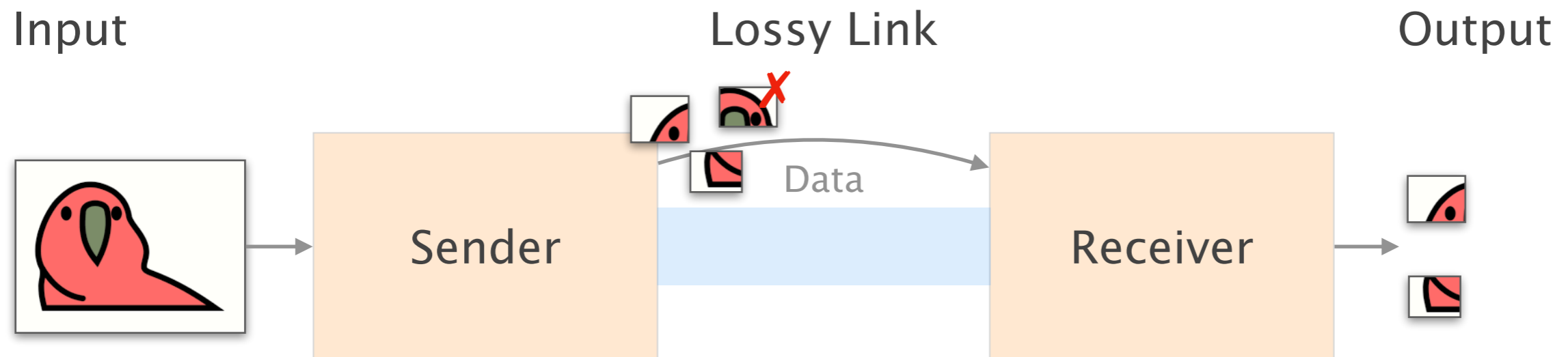
# Implement your own Reliable Transport Protocol



# Implement your own Reliable Transport Protocol

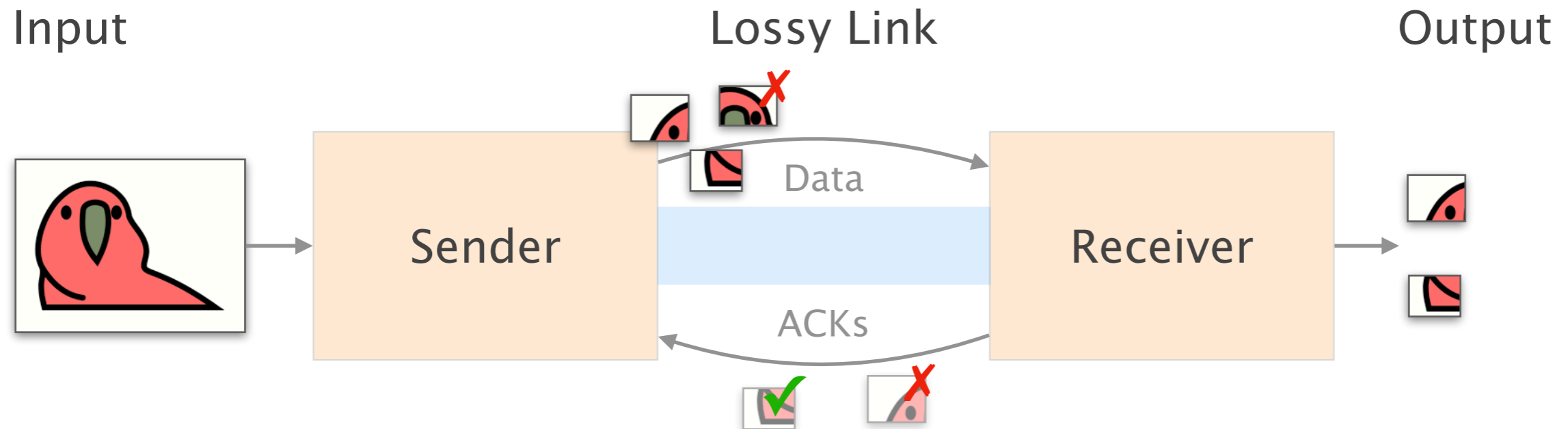


# Implement your own Reliable Transport Protocol

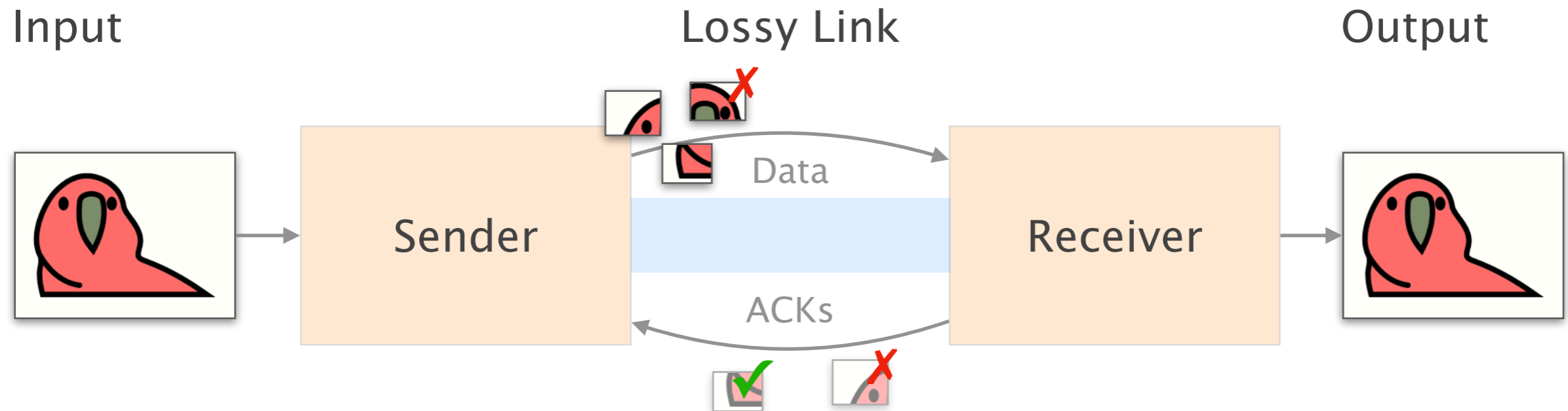




# Implement your own Reliable Transport Protocol



# Implement your own Reliable Transport Protocol



# The Go-Back-N Protocol

# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions

# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions

Sender

Receiver

# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions

Sender

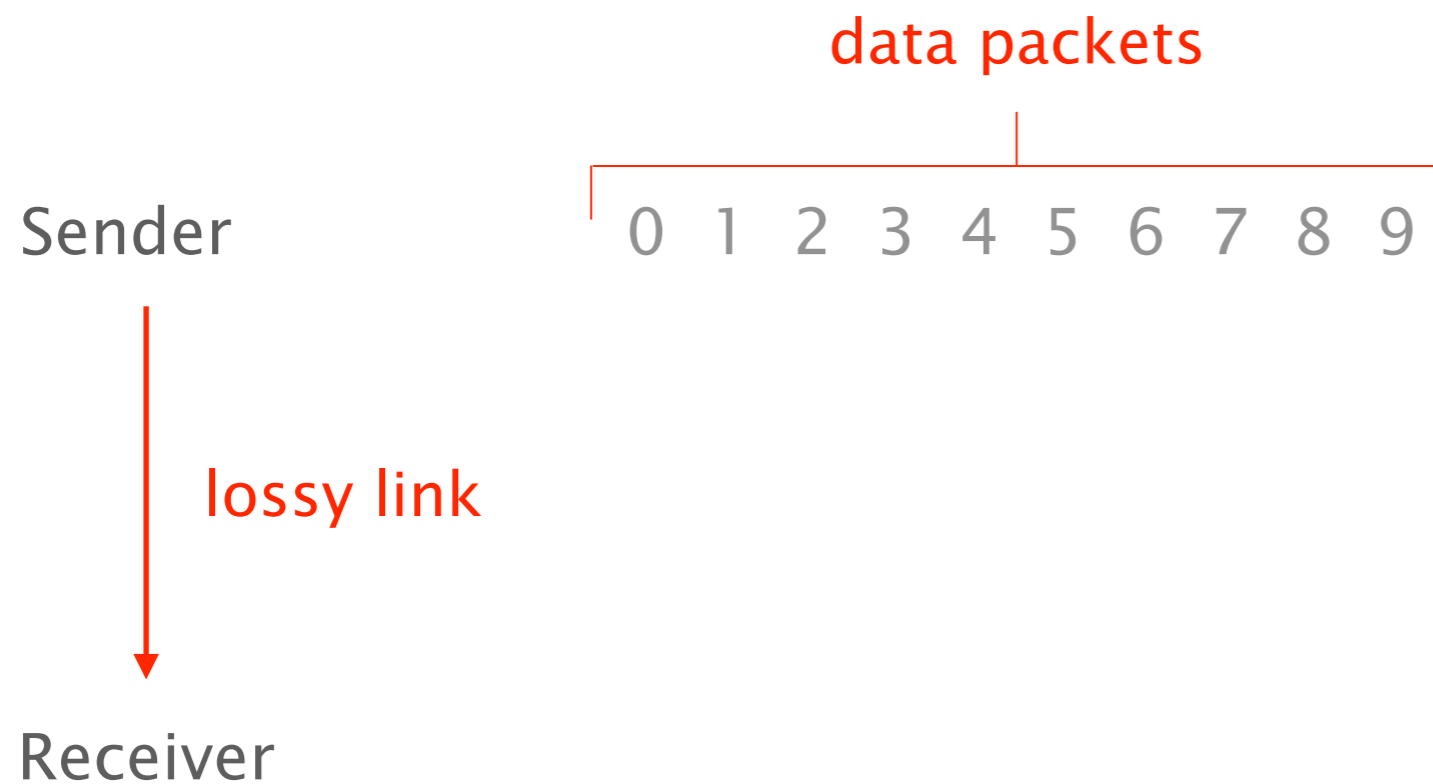


lossy link

Receiver

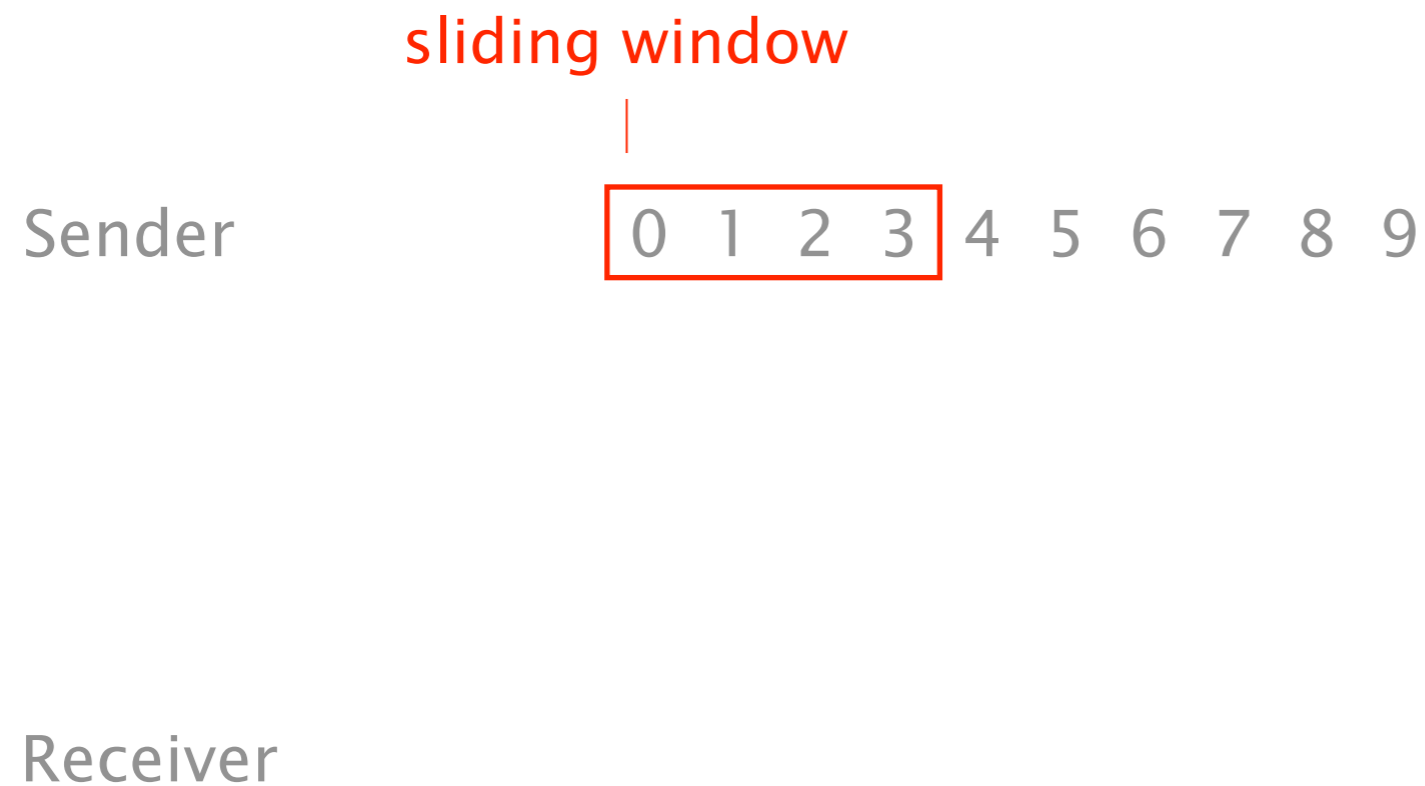
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



# The Go-Back-N Protocol

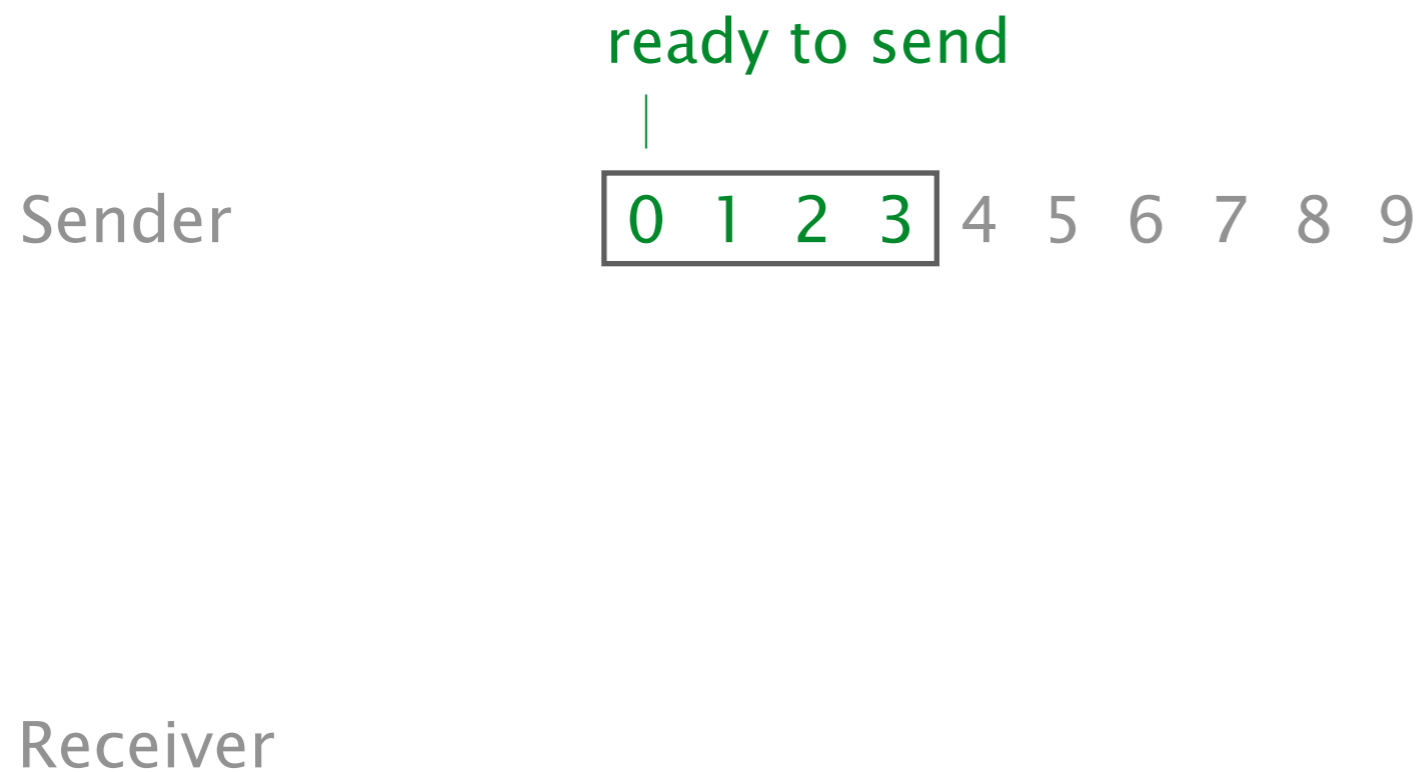
a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions





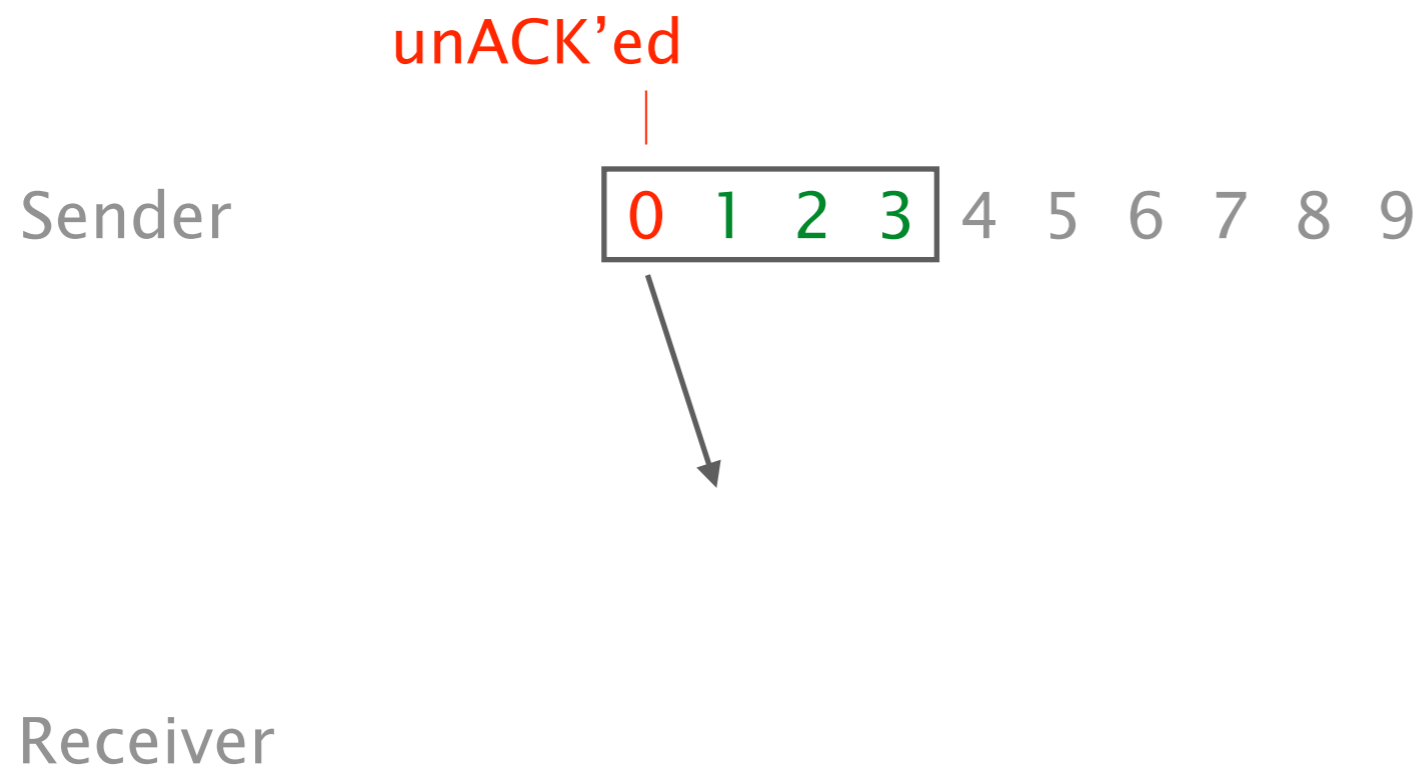
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



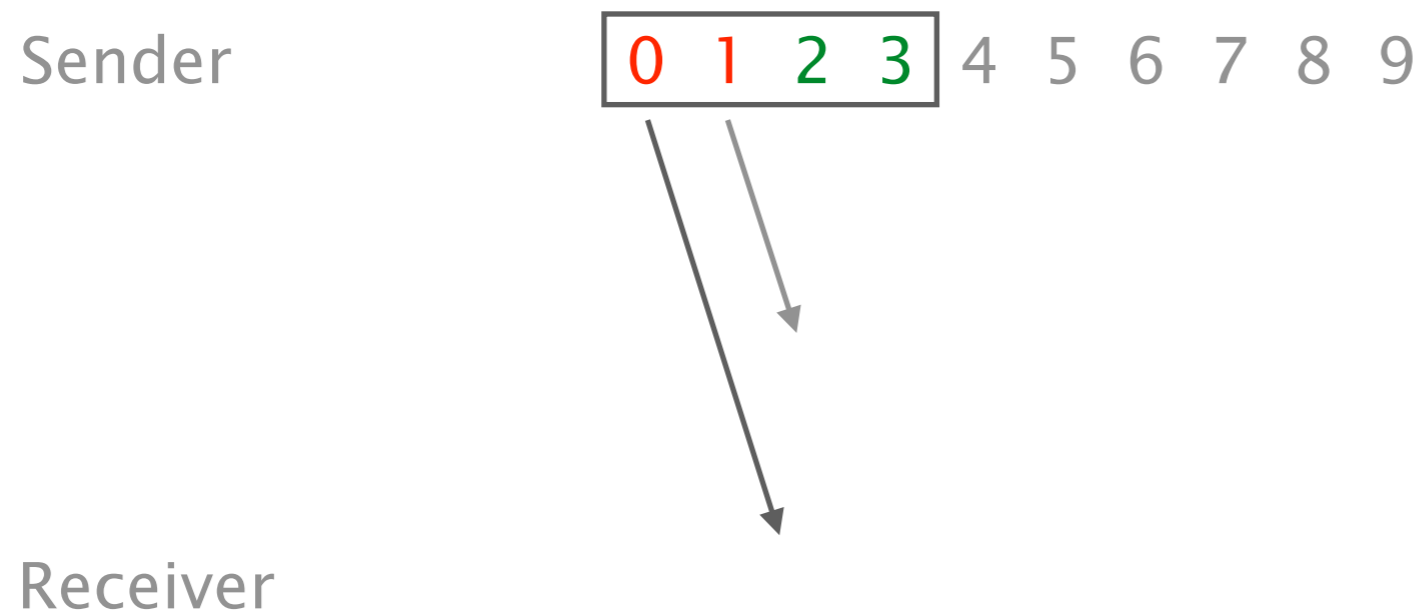
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



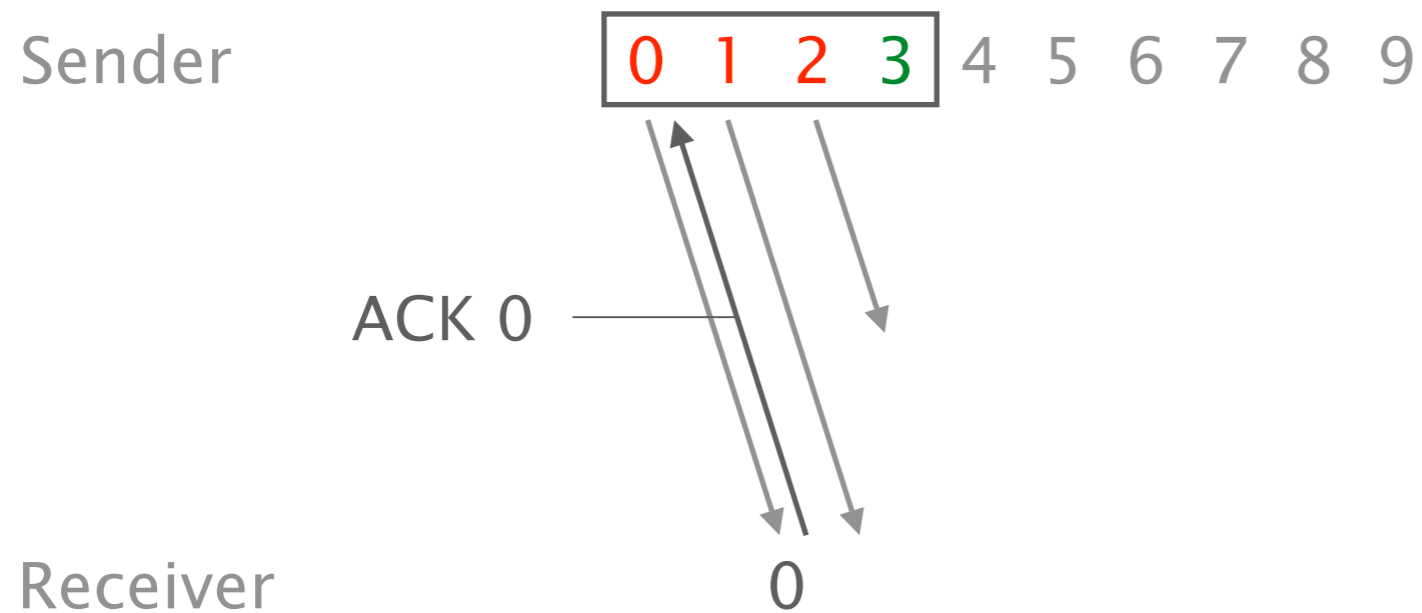
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



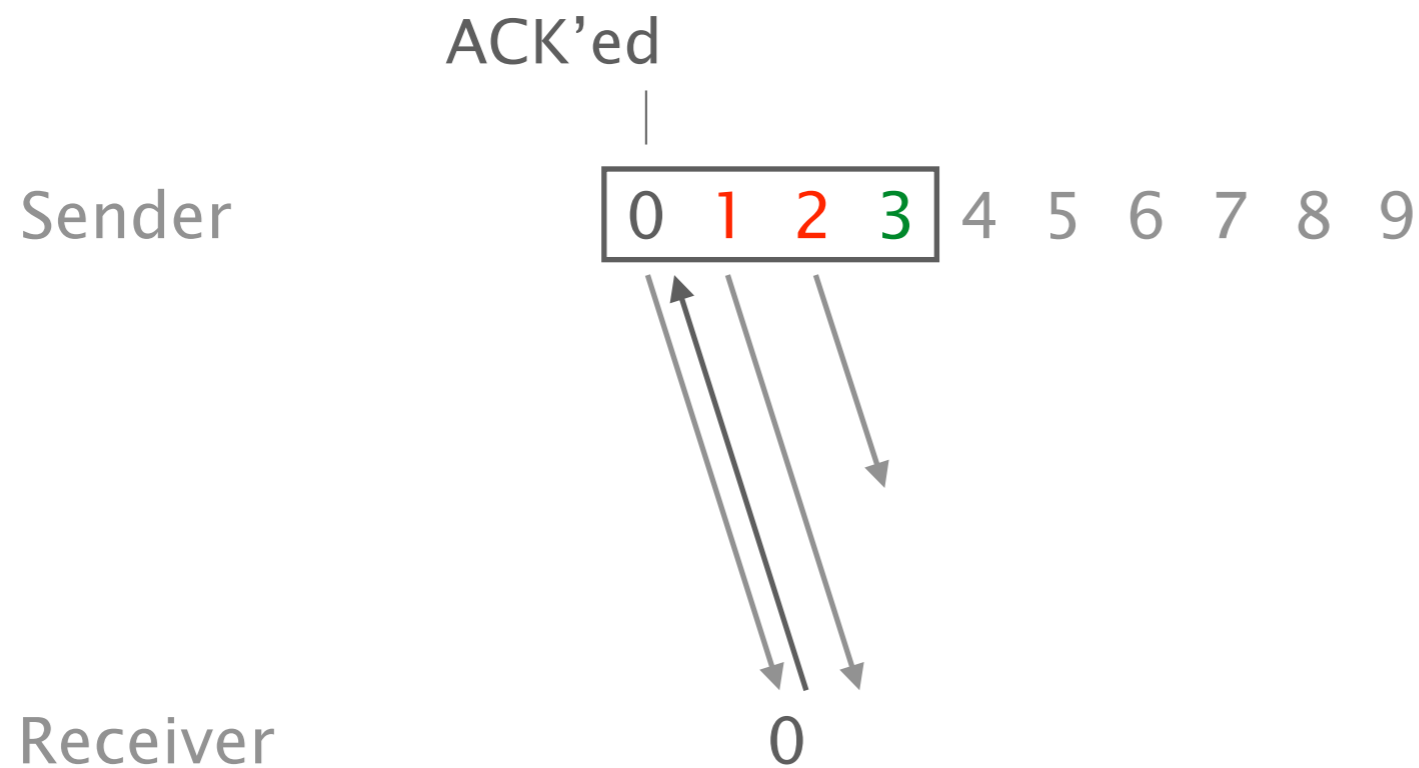
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



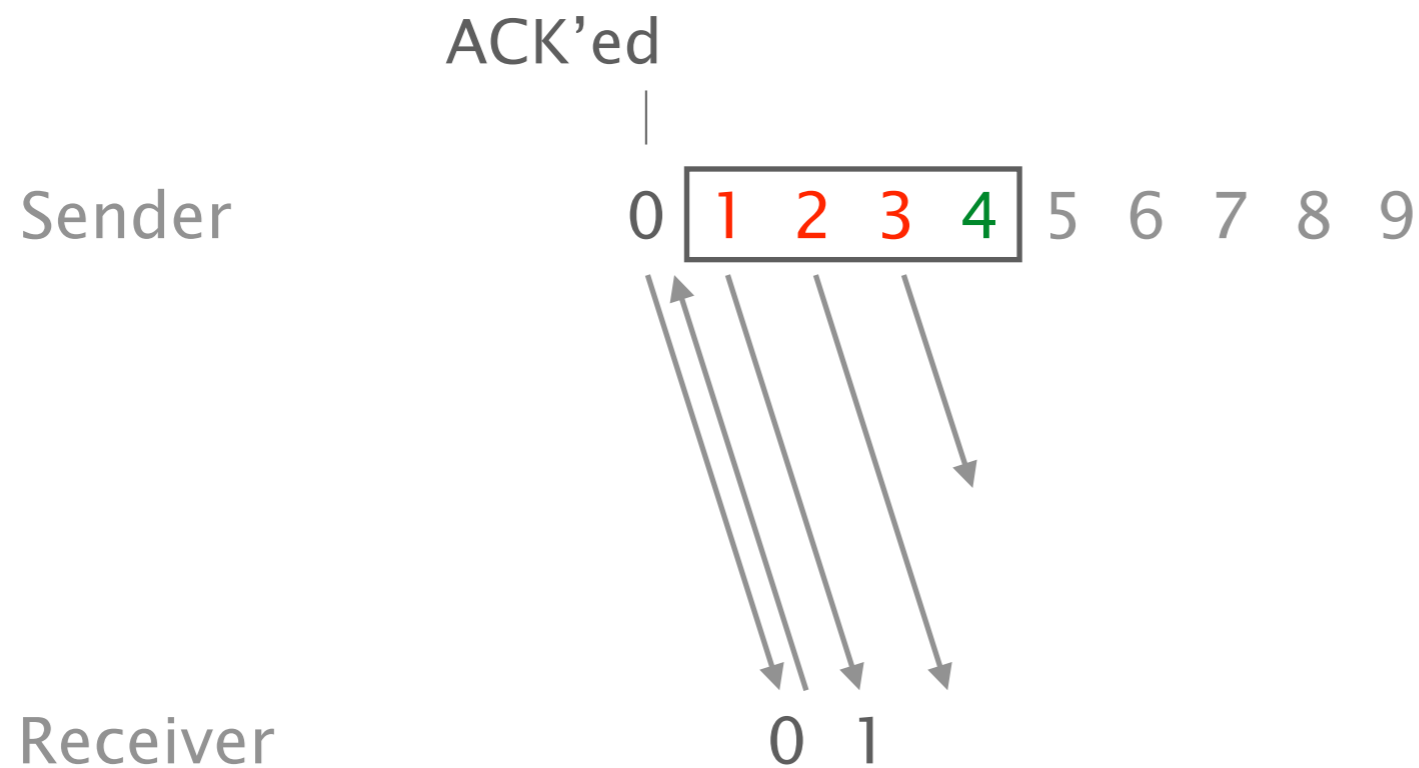
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



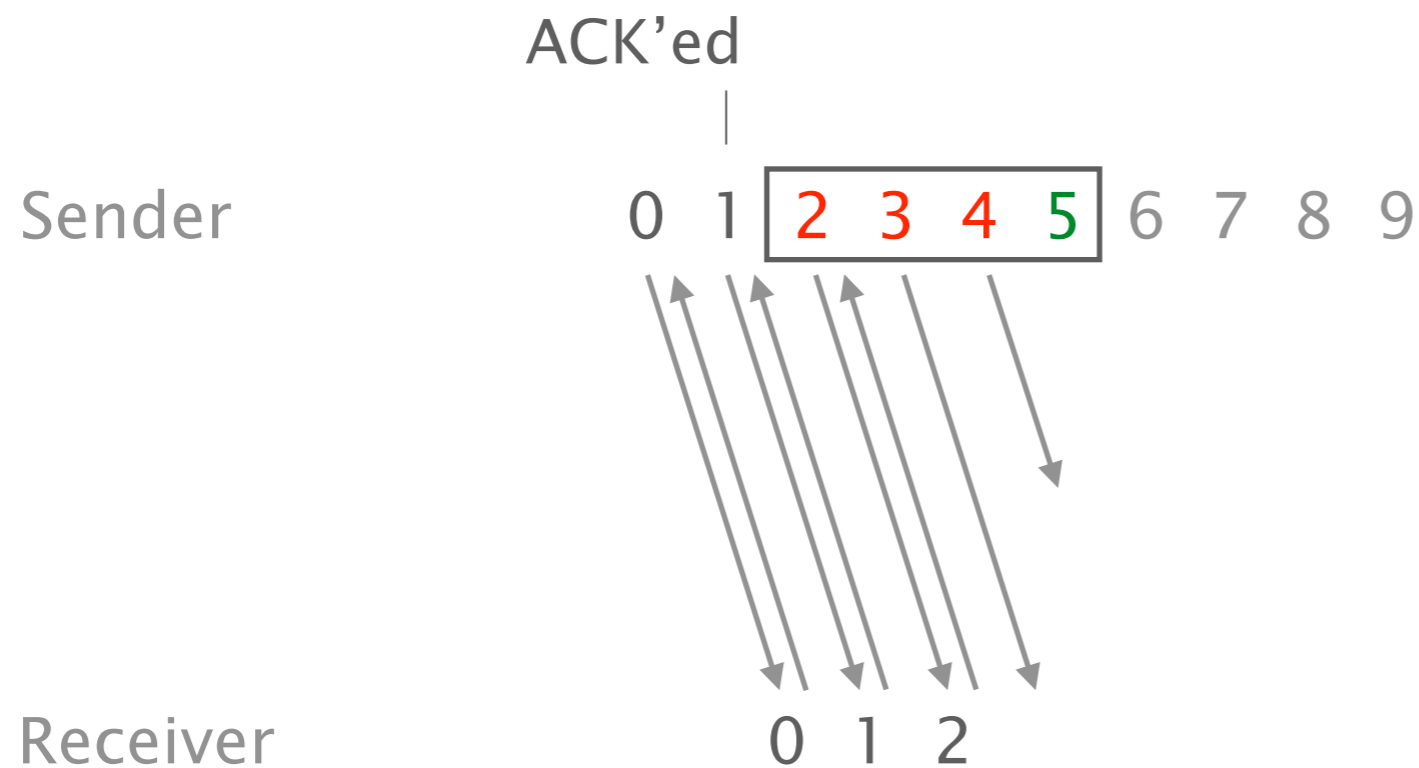
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



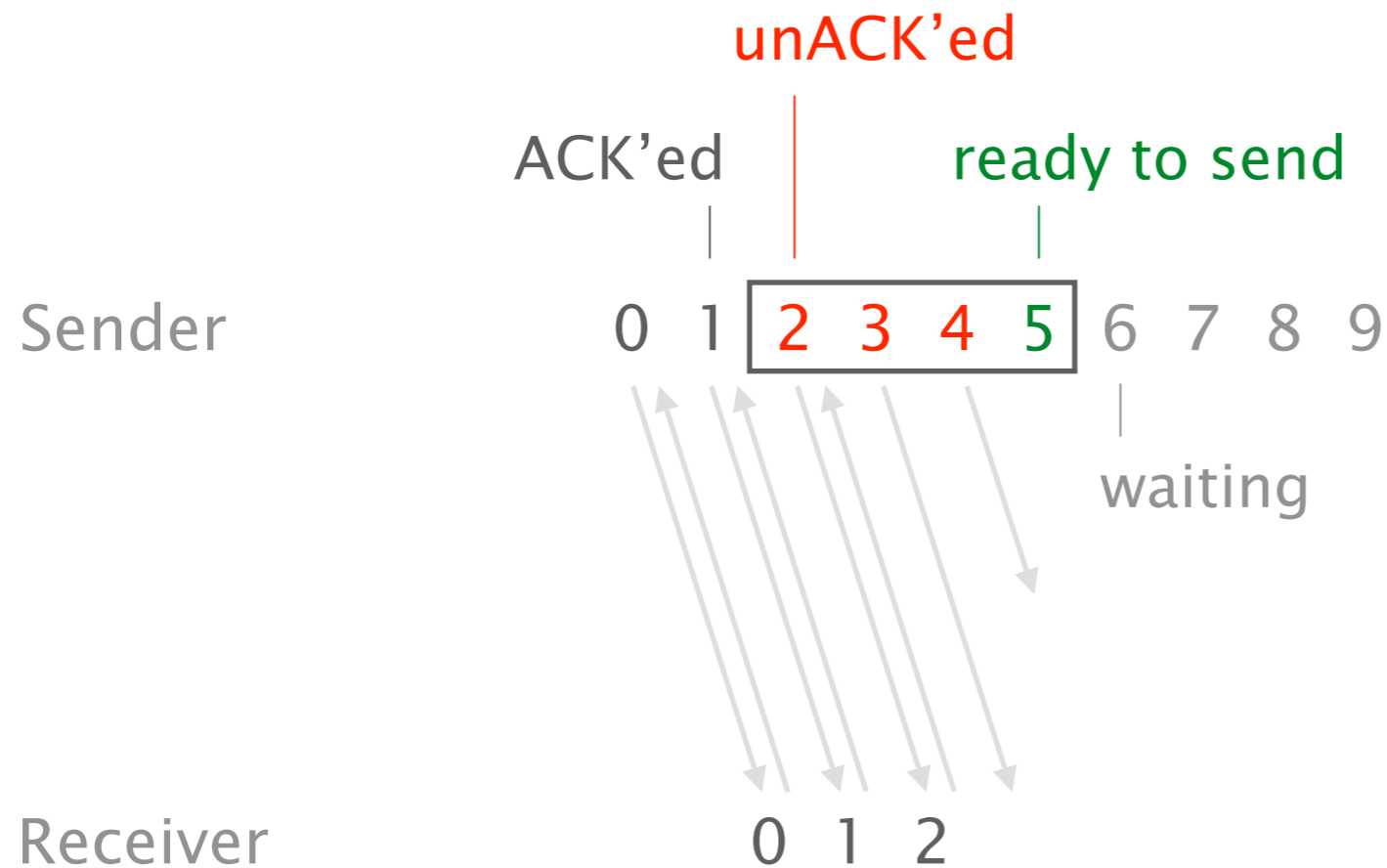
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions





# The Go-Back-N Protocol

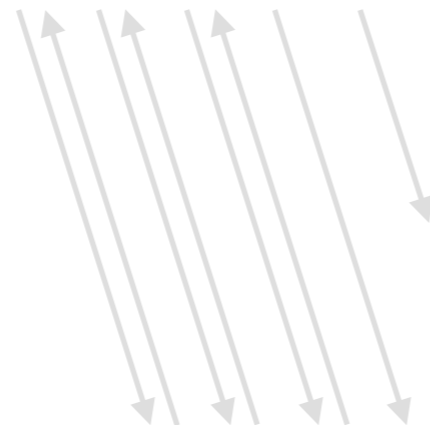
a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions

Sender

0 1 2 3 4 5 6 7 8 9

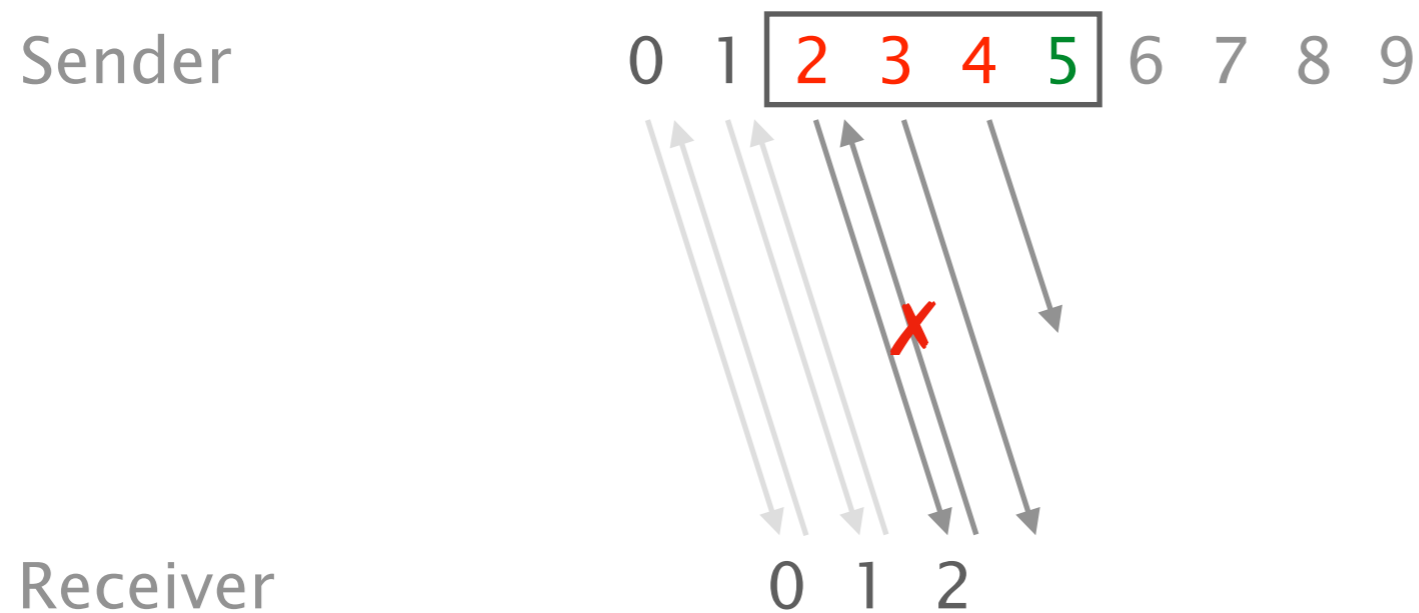
Receiver

0 1 2



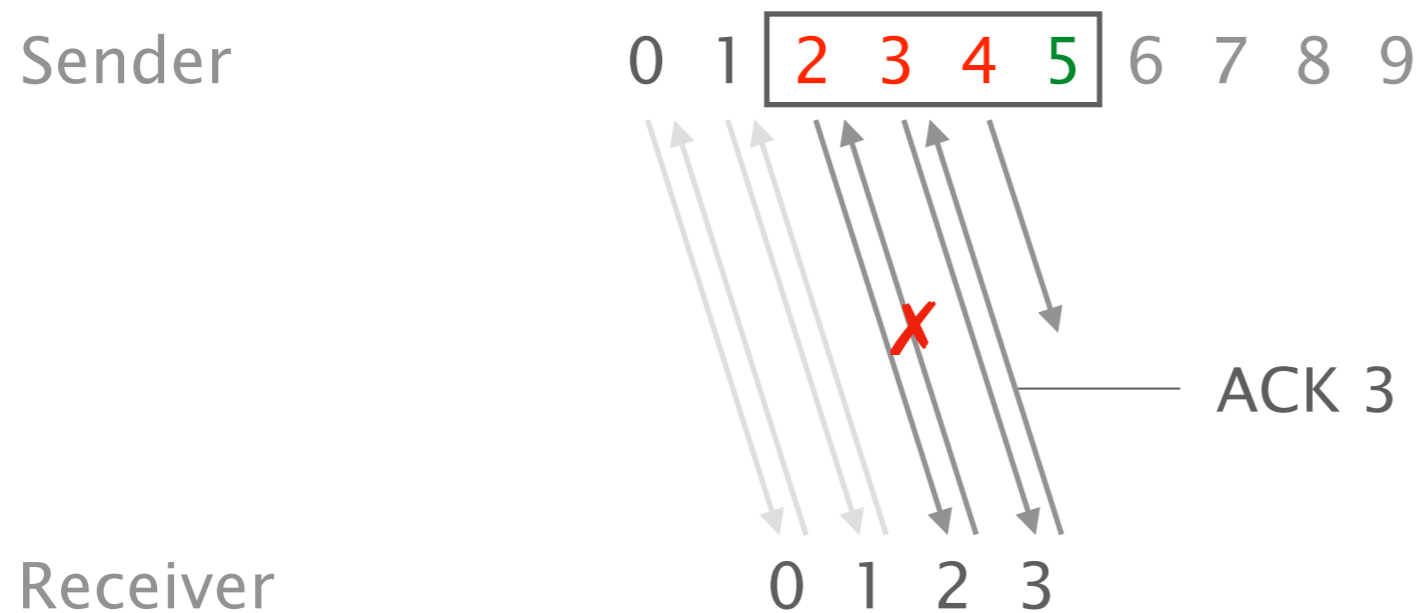
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



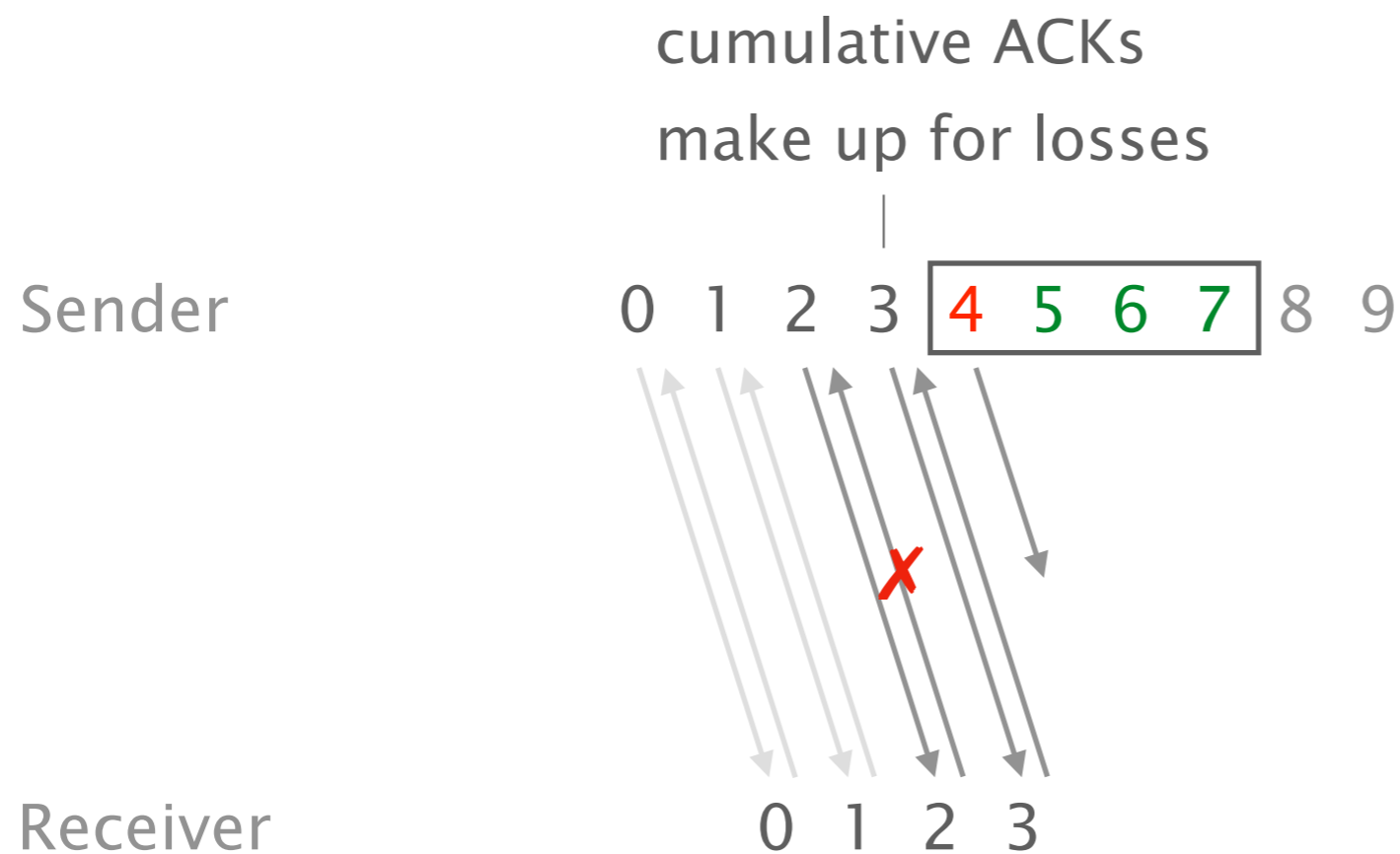
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



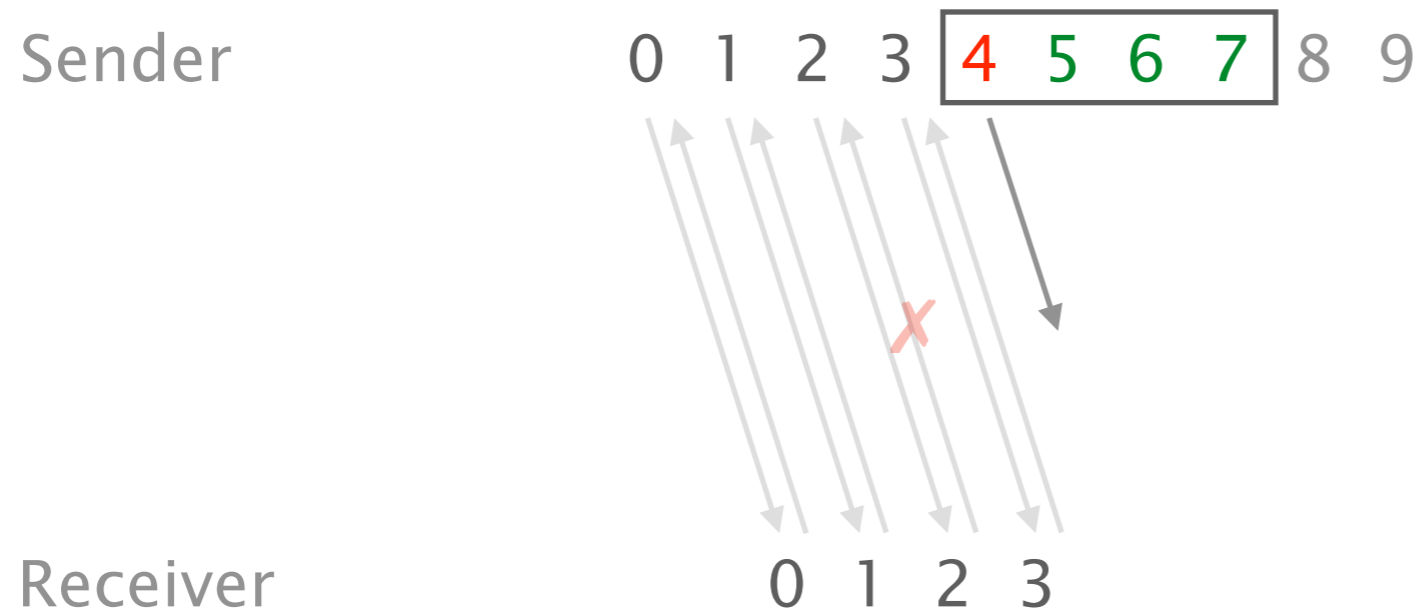
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



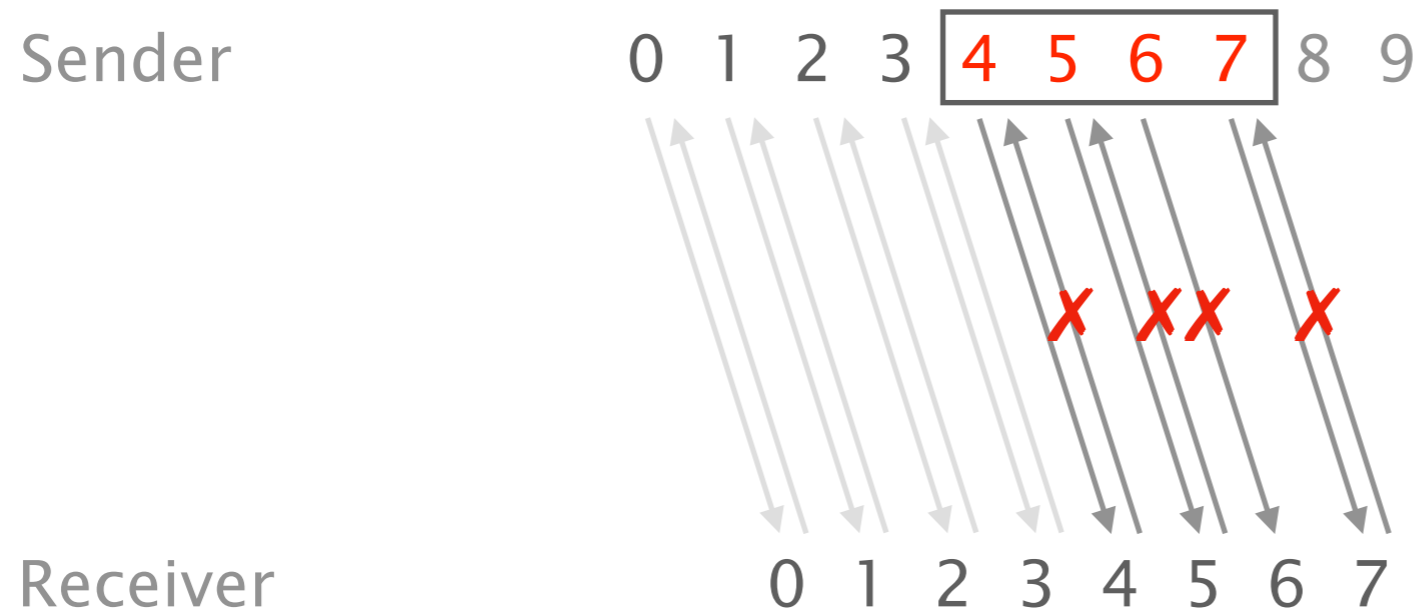
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



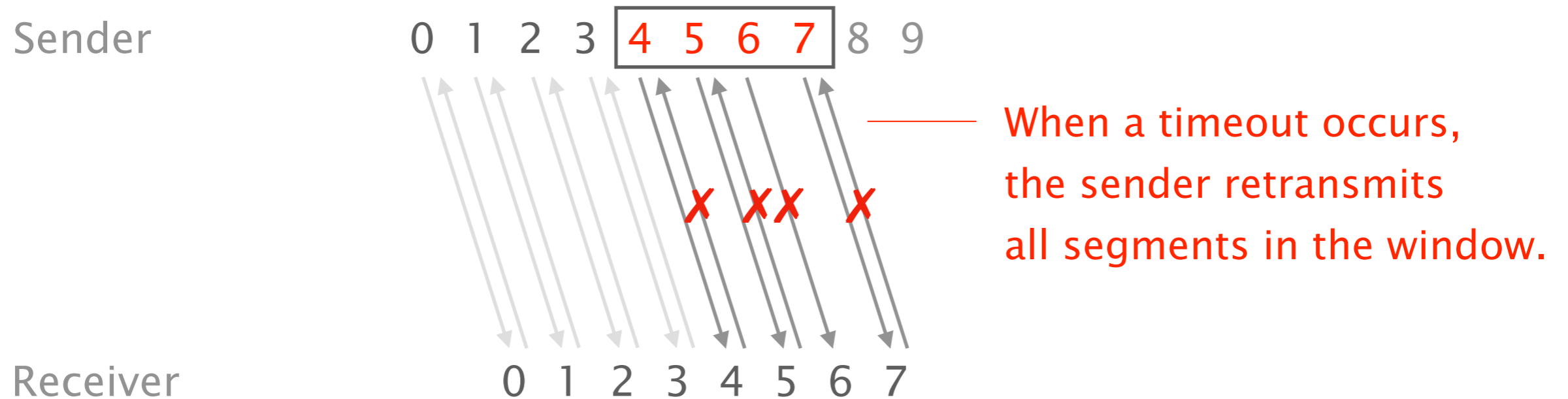
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



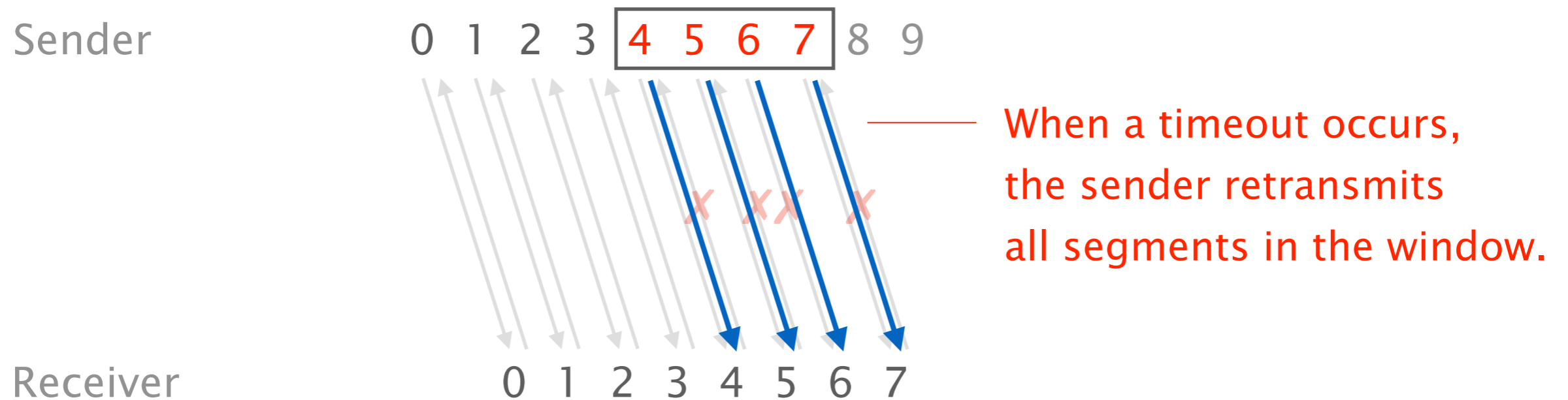
# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions



# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions





# The Go-Back-N Protocol

a simple reliable transport protocol with  
a sliding window, cumulative ACKs, timeouts and retransmissions

Sender



Receiver



# Reliable Transport Project Assignment

- Part 1      Simple Go-Back-N implementation  
Retransmit all packets after a timeout
- Part 2      Support for Selective Repeat  
Fast retransmission after repeated ACKs
- Part 3      Support for Selective Acknowledgements (SACK)  
SACK contains blocks of correctly received segments
- Bonus      Congestion Control

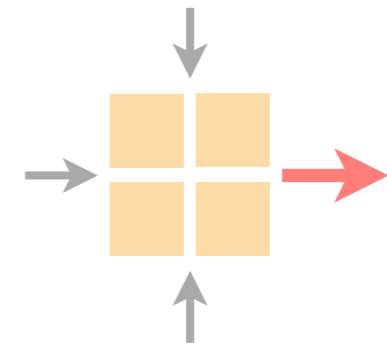
Don't worry, we provide you with a code skeleton

As always:

Ask your questions on **Slack (#transport\_project)**  
or during the **exercise** and **Q&A sessions**

# Communication Networks

## Exercise 10



Wrap-up of the routing project

Intro to the reliable transport project

**Intro to Python and Git**

Current assignment

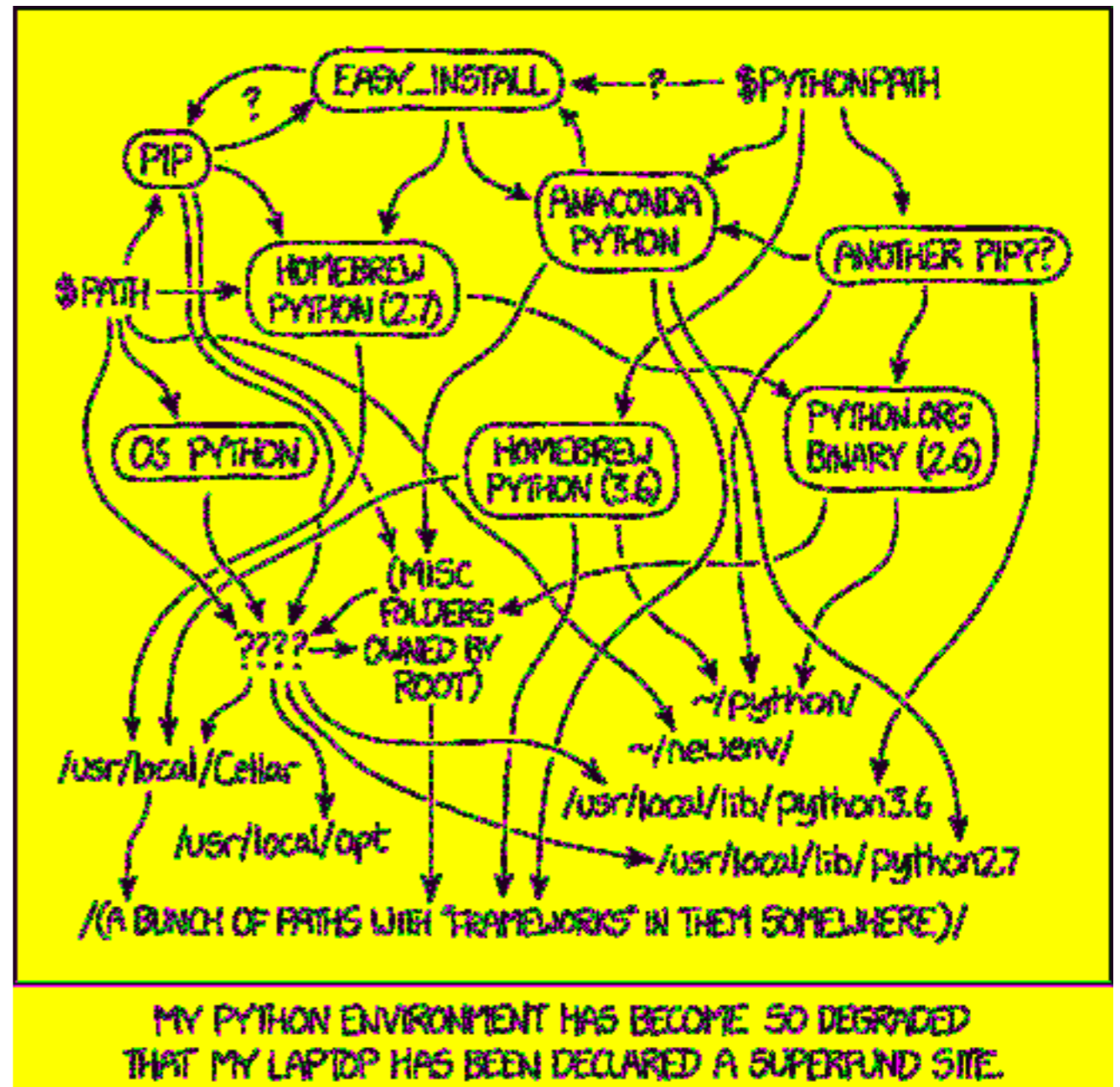
# **Python Development**

**And a bit of Git**

# Contents

- Python
- Integrated Development Environments (IDEs)
- Version Control with Git

# Python



<https://xkcd.com/353/>



# Python Installation

## Windows

[realpython.com/installing-python](https://realpython.com/installing-python)

## Ubuntu

```
sudo apt-get install python3.7 python3-pip \  
python3.7-venv
```

## Mac OSX

```
brew install python
```

# Python

## Getting started

```
$ python3.7
>>> print("Hello World!")
Hello World!
```

Familiarise yourself with Python before you start.

### **Beginners Guide**

[learnpython.org](http://learnpython.org)

### **Advanced/Refresh Guide**

[learnxinyminutes.com/docs/python3](http://learnxinyminutes.com/docs/python3)

```
8 - 1 # => 7
10 * 2 # => 20
35 / 5 # => 7.0

# Integer division rounds down for both positive and negative numbers.
5 // 3 # => 1
-5 // 3 # => -2
5.0 // 3.0 # => 1.0 # works on floats too
-5.0 // 3.0 # => -2.0
```

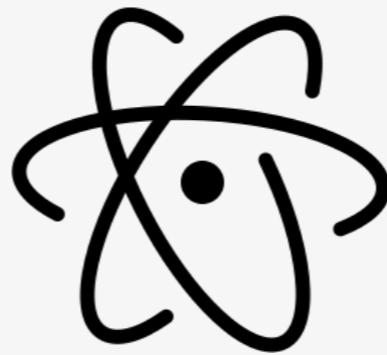
# Python

## From Text Editors to IDEs

### Text Editors



Sublime Text



atom.io

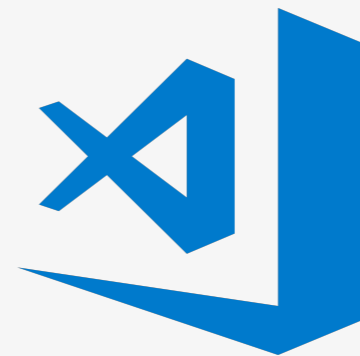


vim

### Integrated Development Environments



JetBrains PyCharm



Visual Studio Code

# Python

## From Text Editors to IDEs



Sublime Text

- Free, no registration required
- Text Editor only
- All platforms

A screenshot of the Sublime Text IDE interface. The window title is 'mfcc.py — mlmcu-project' and it is marked as 'UNREGISTERED'. The left sidebar shows a file explorer for the 'mlmcu-project' directory, with folders like 'audio', 'doc', 'firmware', 'lib', and 'src'. The main editor area displays a Python script with the following code:

```
1 import numpy as np
2 from time import sleep
3 import struct
4 import matplotlib.pyplot as plt
5 from matplotlib import patches
6 from scipy.io import wavfile
7 from scipy.fftpack import dct
8 from tqdm import tqdm
9
10 # Input wav file to use
11 in_wav = 'data/heysnips_true_5k_16b.wav'
12
13 # mel freq. constants -> https://en.wikipedia.org/wiki/Mel_scale
14 MEL_HIGH_FREQUENCY_Q = 1127.0
15 MEL_BREAK_FREQUENCY_HERTZ = 700.0
16
17 def frames(data, frame_length=3, frame_step=1):
18     """
19     Split a data vector into (possibly overlapping) frames
20
21     frame_length: length of each frame
22     frame_step: how many sample to advance the frame each step
```

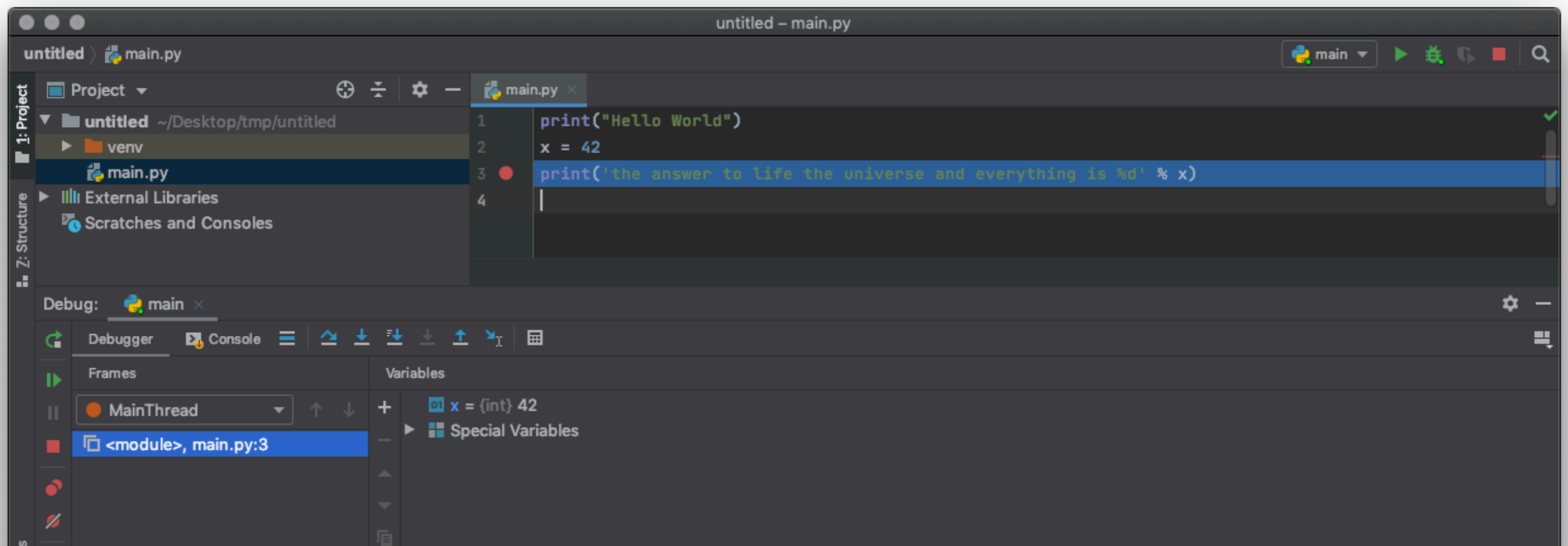
# Python

## From Text Editors to IDEs



JetBrains PyCharm

- Free, open source community edition
- More sophisticated features, such as a debugger
- All platforms



# Python

## Virtual Environments (Advanced topic)

Helps to use correct Python version and packages.

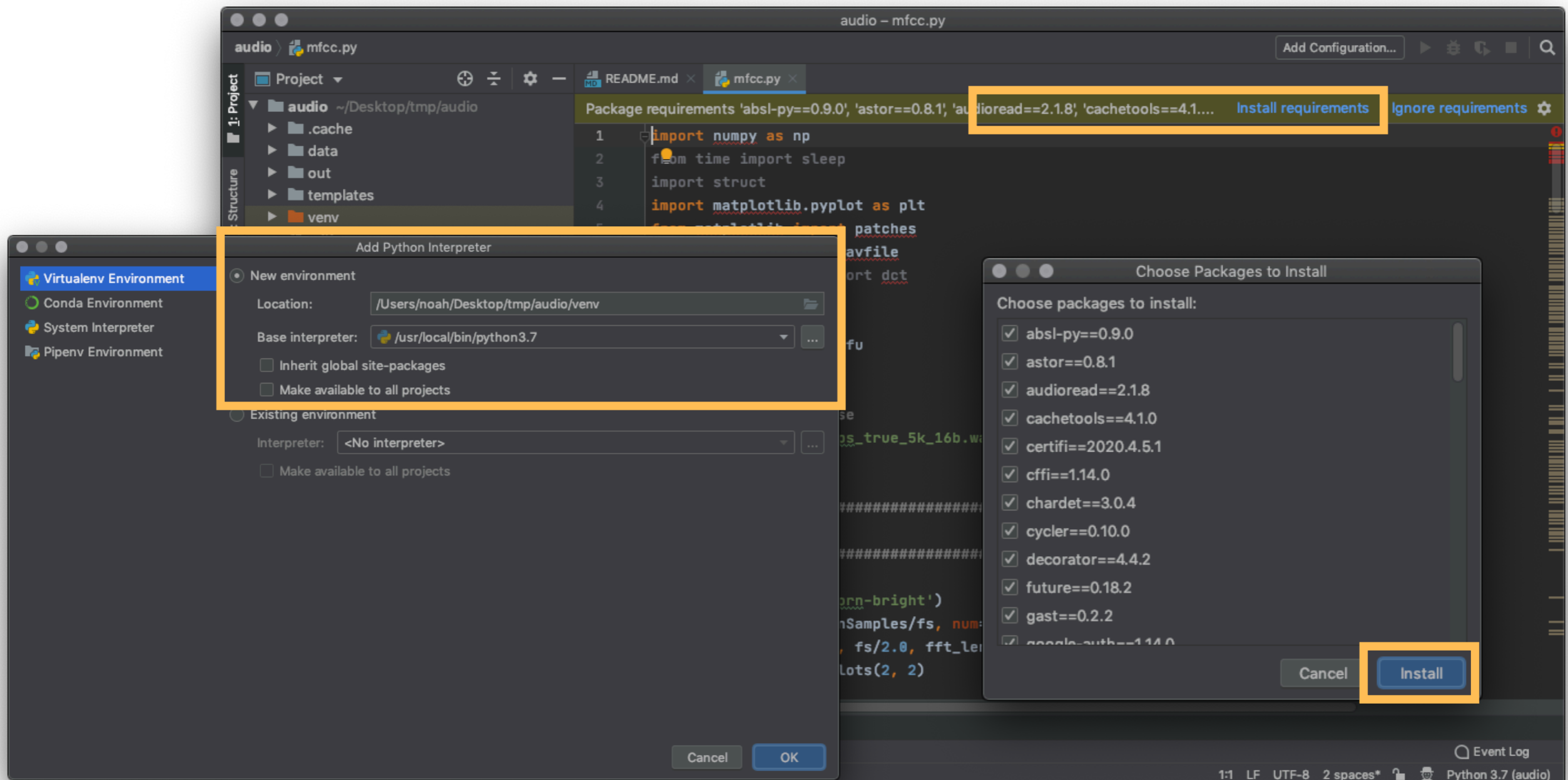
<https://realpython.com/python-virtual-environments-a-primer/>

```
# Create virtual environment (only done to setup)
$ python3.7 -m venv venv
# Activate the environment
$ source venv/bin/activate
# python executable is now used from environment
(venv) $ which python
/Users/noah/venv/bin/python
# Install packages with pip
(venv) $ pip install numpy
# Create list of packages
(venv) $ pip freeze > requirements.txt
# Install all packages from requirements file
(venv) $ pip install -r requirements.txt
# Deactivate environment
(venv) $ deactivate
```

# Python

## Virtual Environments (Advanced topic)

Supported by PyCharm



# Git

## Version Control



<https://xkcd.com/1597/>



# Git

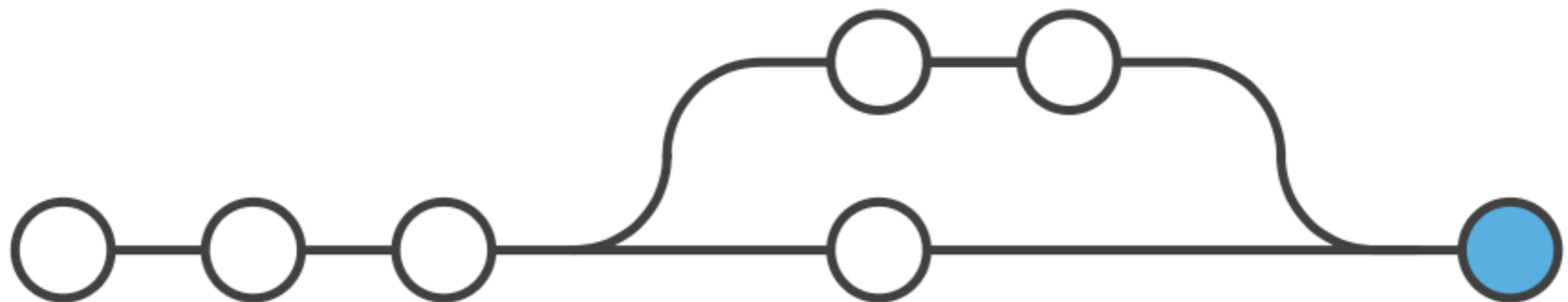
## Tracks Changes in your Code

### Without git

Every collaborator has its **own version of the files**, merging is **manual**, going **back in time is not possible**.

### With git

File **changes are tracked**, merging is **assisted**, **history can be accessed** (and much more)

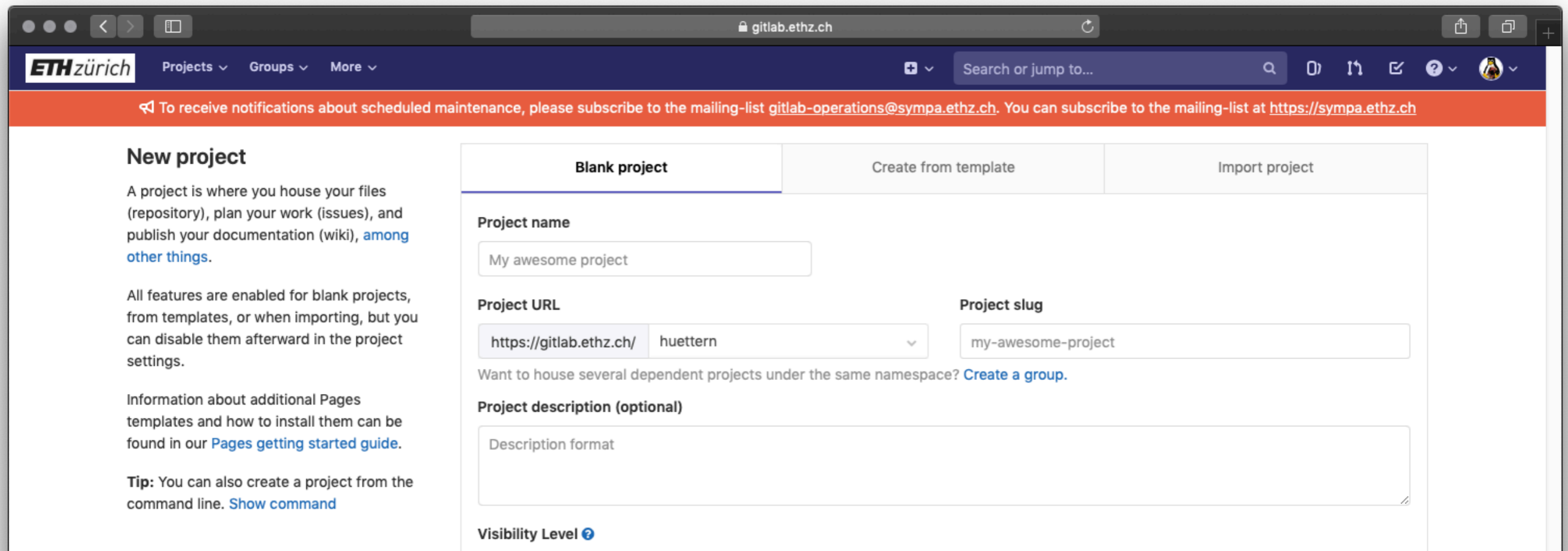


# Git Setup

## 1. Create a repository for your group

<https://gitlab.ethz.ch/projects/new>

You can set the visibility to private (only group members).



The screenshot shows the 'New project' page on the GitLab interface for ethz.ch. The page is divided into three tabs: 'Blank project', 'Create from template', and 'Import project'. The 'Blank project' tab is active. The form includes the following fields:

- Project name:** A text input field containing 'My awesome project'.
- Project URL:** A dropdown menu showing 'https://gitlab.ethz.ch/' and a text input field containing 'huettern'.
- Project slug:** A text input field containing 'my-awesome-project'.
- Project description (optional):** A large text area with a placeholder 'Description format'.
- Visibility Level:** A dropdown menu with a question mark icon.

On the left side, there is a 'New project' section with explanatory text and a tip. The tip states: 'Tip: You can also create a project from the command line. [Show command](#)'.

# Git Setup

## 2. Invite group members

Settings → Members

Set role to developer so they can push to non-protected branches, the `master` branch is protected.

The screenshot shows the GitLab web interface for a project named 'git-demo'. The page is titled 'Project members' and includes a sub-header 'Project members' and a description: 'You can invite a new member to git-demo or invite another group.' Below this, there are two tabs: 'Invite member' (selected) and 'Invite group'. The 'Invite member' tab contains a form with a text input field labeled 'GitLab member or Email address' containing the text 'buehlert', a dropdown menu labeled 'Choose a role permission' set to 'Developer', and a link 'Read more about role permissions'. The 'Access expiration date' field is partially visible at the bottom.

# Git

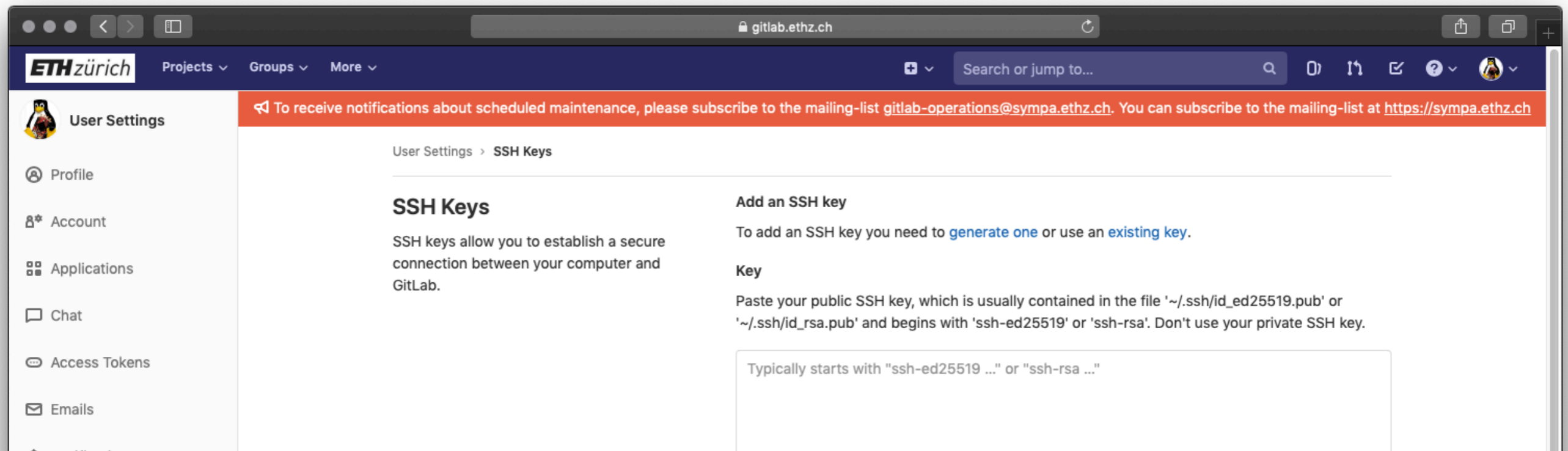
## Setup

### 3. Create SSH key and add it to Gitlab

<https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html>

```
$ ssh-keygen
```

This allows you to access the repository from the console.

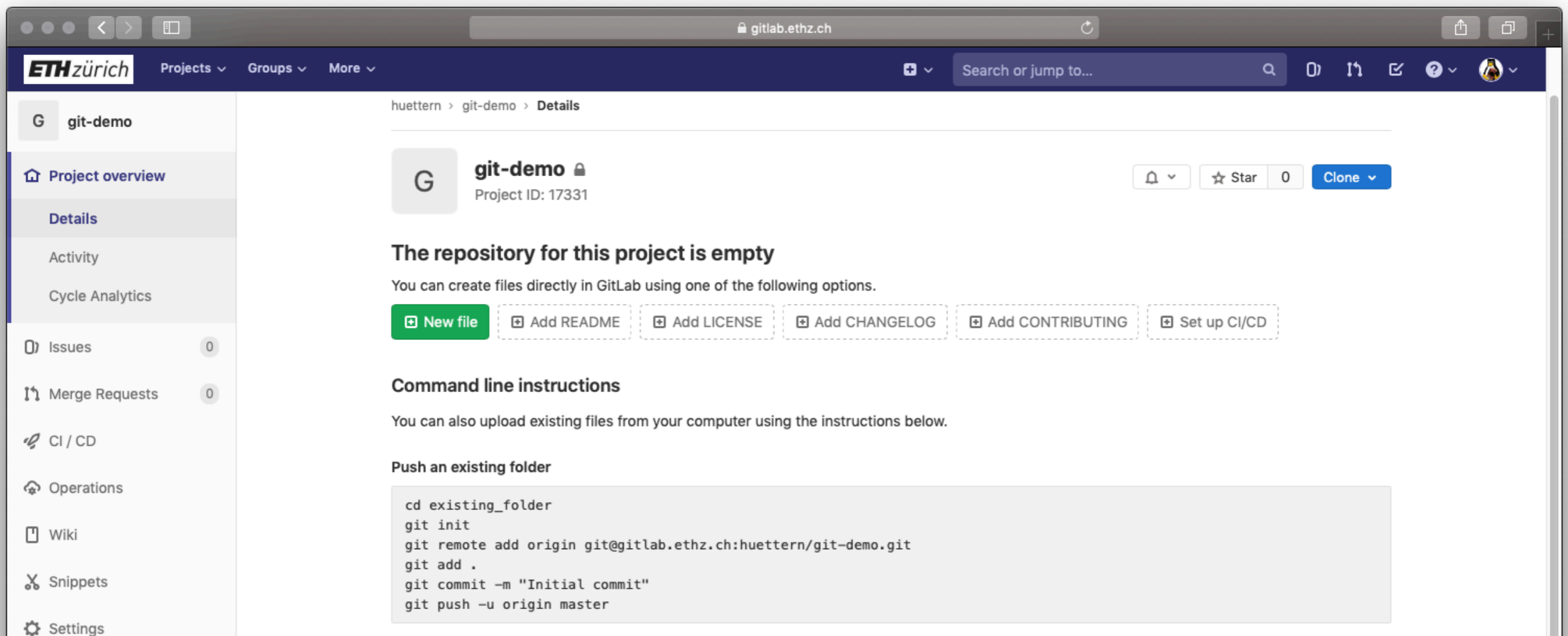


The screenshot shows a web browser window displaying the GitLab user settings page for SSH Keys. The browser's address bar shows 'gitlab.ethz.ch'. The page header includes the ETH zürich logo and navigation links for Projects, Groups, and More. A search bar is visible in the top right. A red notification banner at the top of the page reads: 'To receive notifications about scheduled maintenance, please subscribe to the mailing-list [gitlab-operations@sympa.ethz.ch](mailto:gitlab-operations@sympa.ethz.ch). You can subscribe to the mailing-list at <https://sympa.ethz.ch>'. The main content area is titled 'User Settings > SSH Keys'. It features a section for 'SSH Keys' with a description: 'SSH keys allow you to establish a secure connection between your computer and GitLab.' To the right, there is a section for 'Add an SSH key' with instructions: 'To add an SSH key you need to [generate one](#) or use an [existing key](#).' Below this, a 'Key' section explains: 'Paste your public SSH key, which is usually contained in the file '~/.ssh/id\_ed25519.pub' or '~/.ssh/id\_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.' A text input field is provided for pasting the key, with a placeholder text: 'Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."'. The left sidebar contains navigation options: User Settings, Profile, Account, Applications, Chat, Access Tokens, and Emails.

# Git Setup

## 4. Upload the project files to Gitlab

Go to the repository (Projects → repository name) and follow the instructions for *Push an existing folder*.



The screenshot shows the GitLab interface for a repository named 'git-demo'. The page title is 'huettern > git-demo > Details'. The repository is currently empty, and the user is prompted to create files directly in GitLab using one of the following options:

- New file
- Add README
- Add LICENSE
- Add CHANGELOG
- Add CONTRIBUTING
- Set up CI/CD

Below these options, the 'Command line instructions' section provides the following instructions for pushing an existing folder:

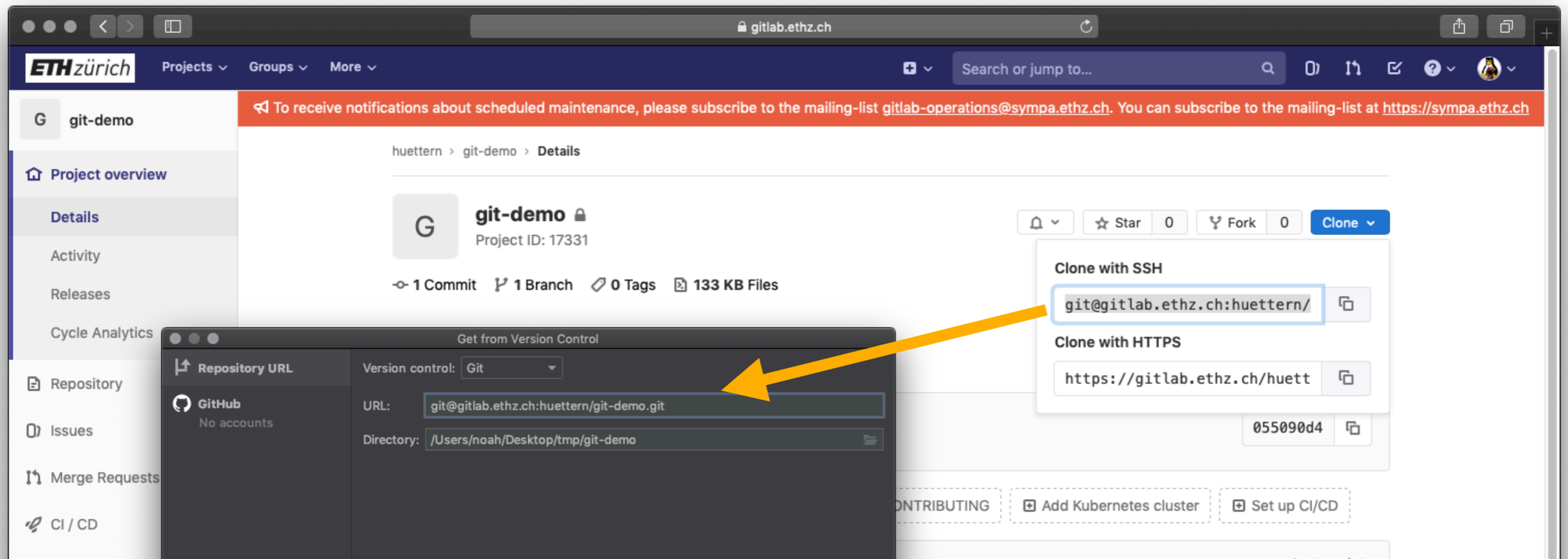
```
cd existing_folder
git init
git remote add origin git@gitlab.ethz.ch:huettern/git-demo.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

# Git Setup

## 5. Download the repository to your local machine

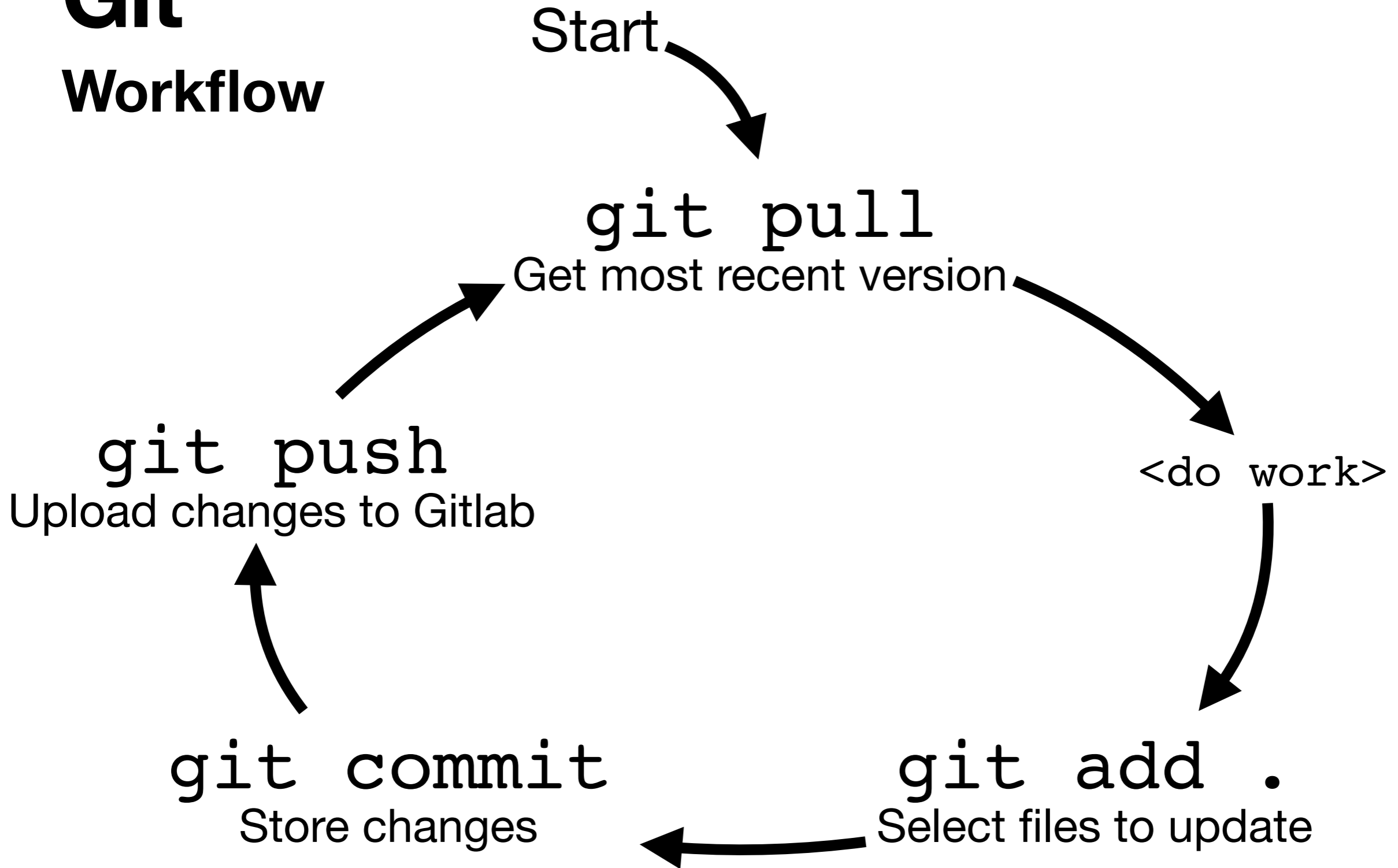
This way you can work on your machine without VM connection

```
$ git clone <link_to_repo>
```



The image shows a screenshot of a web browser displaying a GitLab repository page for 'git-demo' on 'gitlab.ethz.ch'. The repository details include 'Project ID: 17331', '1 Commit', '1 Branch', '0 Tags', and '133 KB Files'. A 'Clone' button is visible, which has opened a dropdown menu showing two options: 'Clone with SSH' and 'Clone with HTTPS'. The SSH URL 'git@gitlab.ethz.ch:huettern/' is highlighted with a blue box. Below the repository page, a 'Get from Version Control' dialog box is open, showing the 'Repository URL' field with the same SSH URL 'git@gitlab.ethz.ch:huettern/git-demo.git' entered. A yellow arrow points from the highlighted SSH URL in the dropdown menu to the 'Repository URL' field in the dialog box. The dialog box also shows 'Version control: Git' and 'Directory: /Users/noah/Desktop/tmp/git-demo'.

# Git Workflow



# Git

## Workflow

```
# Download latest changes from Gitlab
$ git pull
# Do work on files...
$ vim main.py
# Show what has changed
$ git status
# Add the files you want to update
$ git add main.py
# Store changes in history with a short description
$ git commit -m "very important bug fix"
# Upload the changes to Gitlab
$ git push
```



# Git

## Workflow

The screenshot displays an IDE window titled "audio - mfcc.py". The top toolbar includes a "speechrecognizer" dropdown and a "Git" status indicator. The left sidebar shows "Local Changes" with a "Default Changelist" containing "mfcc.py". The main editor area shows the following code for "mfcc.py":

```
5 from matplotlib import patches
6 from scipy.io import wavfile
7 from scipy.fftpack import dct
8 from tqdm import tqdm
9
10 import mfcc_utils as mfu
11
12
13 # Input wav file to use
```

The bottom pane shows the Git commit history for "speechrecognizer.py". The commit list includes:

- start playing around with cmplx magnitude! (master) Noah Huetter Today 08:17
- real fft also working! (origin/master) Noah Huetter Yesterday 19:33
- FFT WOKRING YAY Noah Huetter Yesterday 19:32
- Add reliability to sending data Noah Huetter Yesterday 17:41
- starting mfcc mcu implementation Noah Huetter 20.04.20, 21:34
- req ack function, pingpong stil not workin Noah Huetter 20.04.20, 17:20
- data transfer workin, pinpong test not Noah Huetter 20.04.20, 15:52
- MCU to host data exchange working Noah Huetter 20.04.20, 11:59
- revert file for normal op Noah Huetter 20.04.20, 10:42
- store figure Noah Huetter 20.04.20, 10:42
- Add plotting for different MFCC implementations Noah Huetter 20.04.20, 10:41
- add lab on mfcc Noah Huetter 20.04.20, 09:55
- output to sigmoid and binary crossentropy make resul Noah Huetter 20.04.20, 09:24

The bottom status bar shows "12:1 LF UTF-8 2 spaces\* Python 3.7 (verw) master".

# Git

## Tips and Tricks

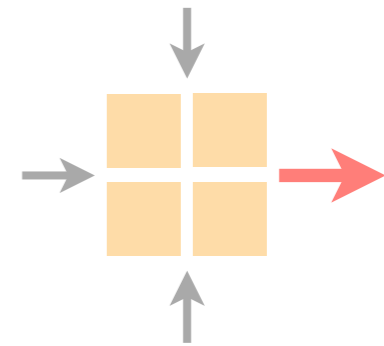
- No branching required for the assignment
- Run the git commands from the correct directory
- Always `pull` before you `push`

Cheat Sheet & Installation Guide

[rogerdudler.github.io/git-guide](https://rogerdudler.github.io/git-guide)

# Communication Networks

## Exercise 10



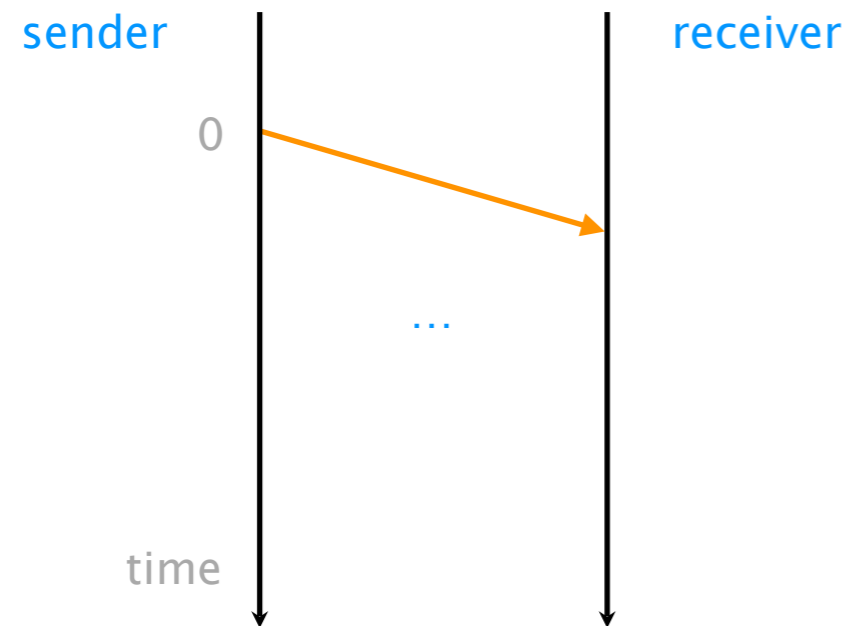
Wrap-up of the routing project

Intro to the reliable transport project

Intro to Python and Git

**Current assignment**

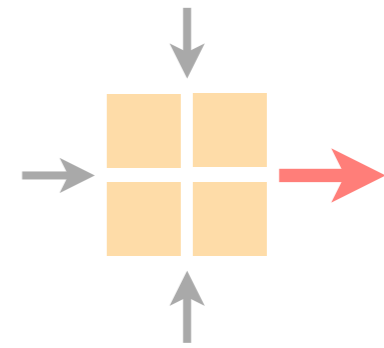
# Task 1: Reliable Transport



Analyze a Go-Back-N transfer of 10 segments (10'000 bits) on a 10Mbps link with a 100ms propagation delay with and without loss

# Communication Networks

## Exercise 10



Wrap-up of the routing project

Intro to the reliable transport project

Intro to Python and Git

Current assignment