Communication Networks

Spring 2020

Rüdiger Birkner, TA

Thomas Holterbach, TA

Noah Hütter, TA

Eric Marty, TA

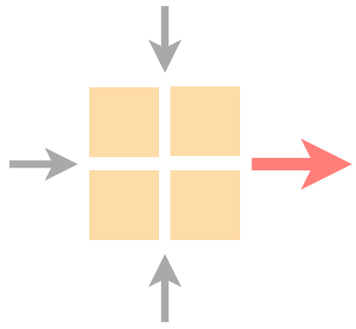
https://comm-net.ethz.ch/

ETH Zürich

April 23 2020

Communication Networks

Exercise 10



Wrap-up of the routing project

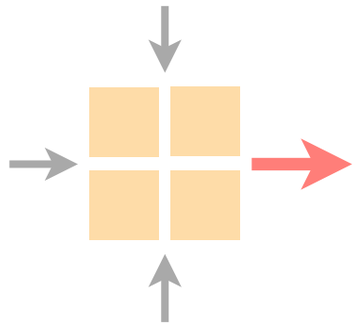
Intro to the reliable transport project

Intro to Python and Git

Current assignment

Communication Networks

Exercise 10



Wrap-up of the routing project

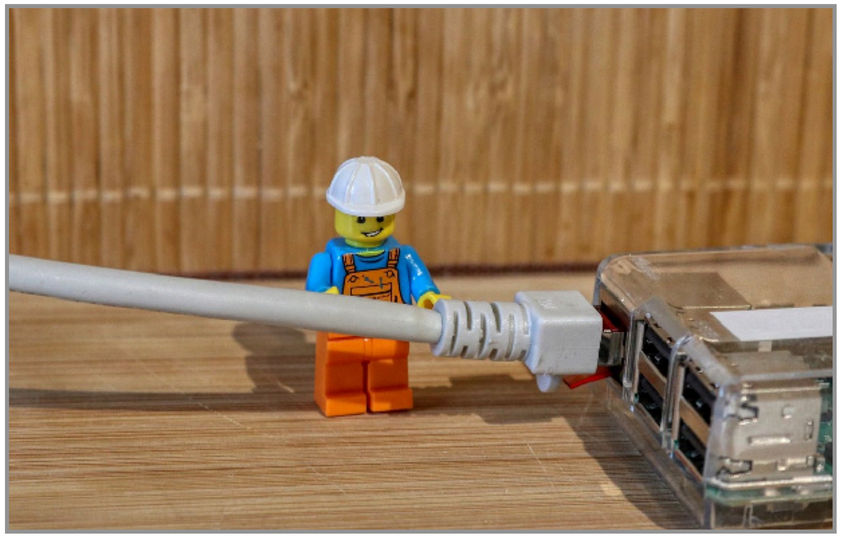
Intro to the reliable transport project

Intro to Python and Git

Current assignment

Communication Networks 2020

How we build a mini-Internet



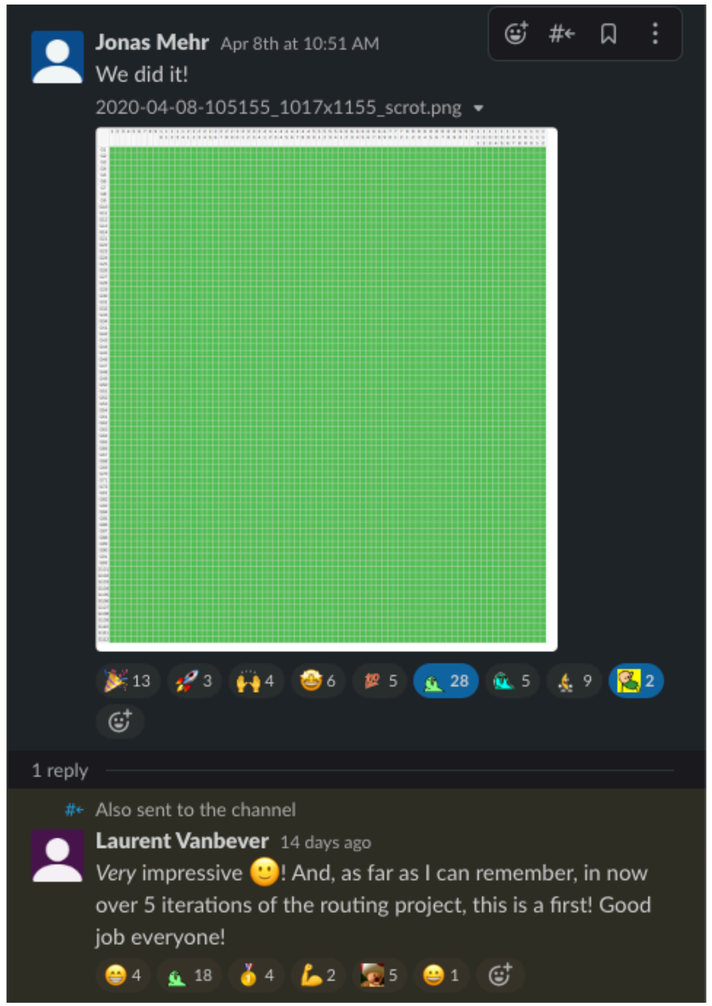
Thomas Holterbach

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

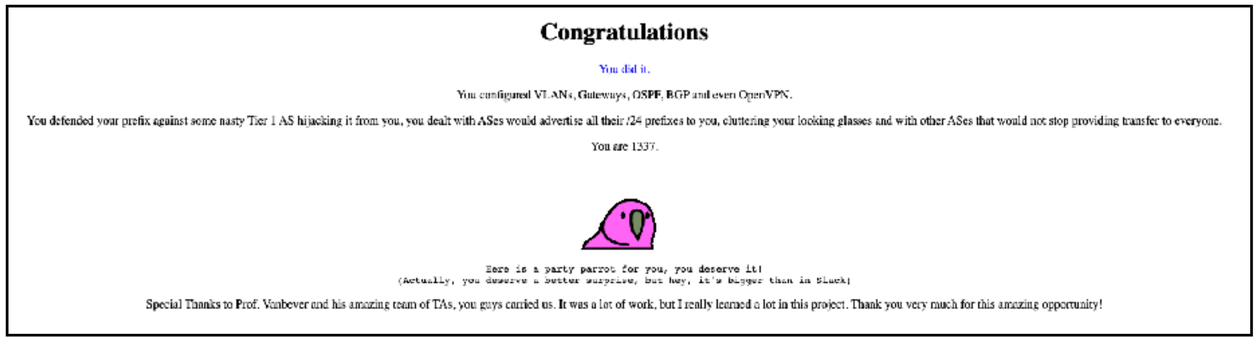
ETH Zurich (D-ITET)

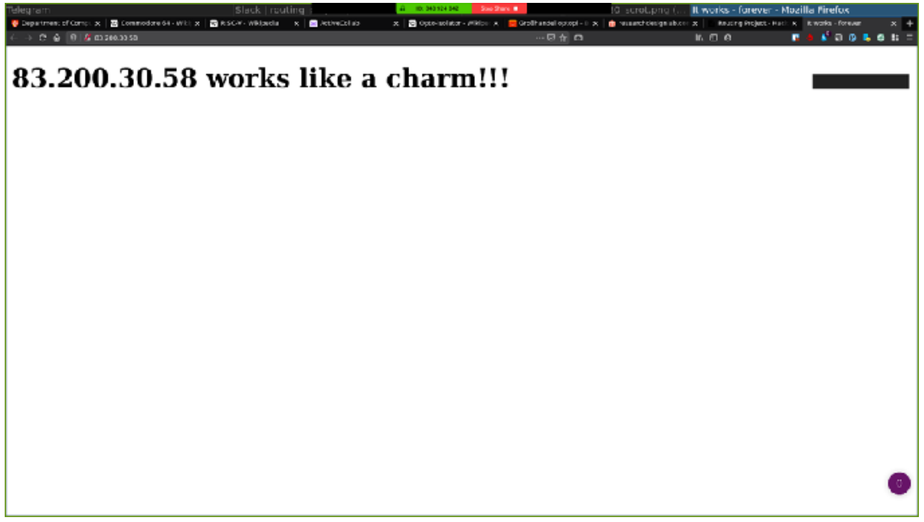
April 23, 2020

You did it: 100% connectivity!

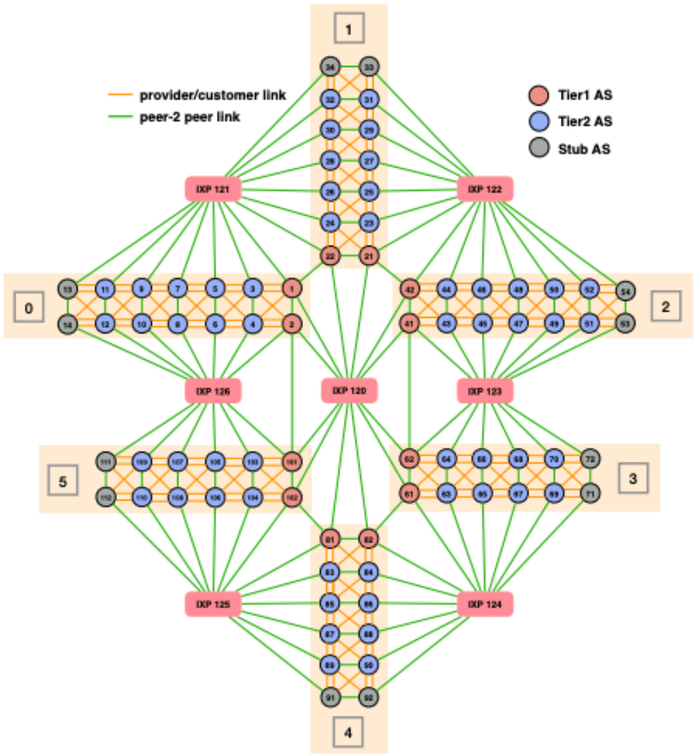


Many of you managed to solve the bonus question!





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

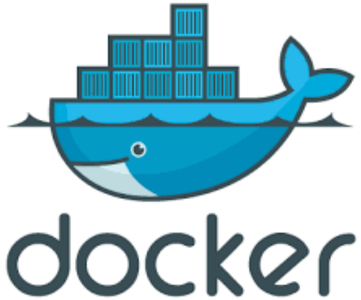


snowball.ethz.ch

We rely on virtualisation

Option #1: virtual machines

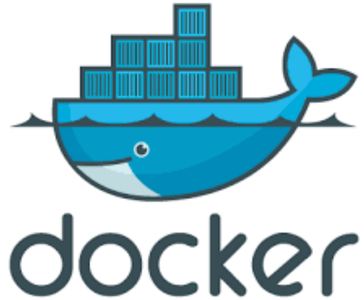
Option #2: linux containers



We rely on virtualisation

Option #1: virtual machines

Option #2: linux containers



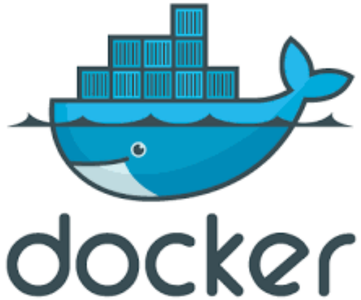
we used VMs between
2016 and 2019

We rely on virtualisation

Option #1: virtual machines

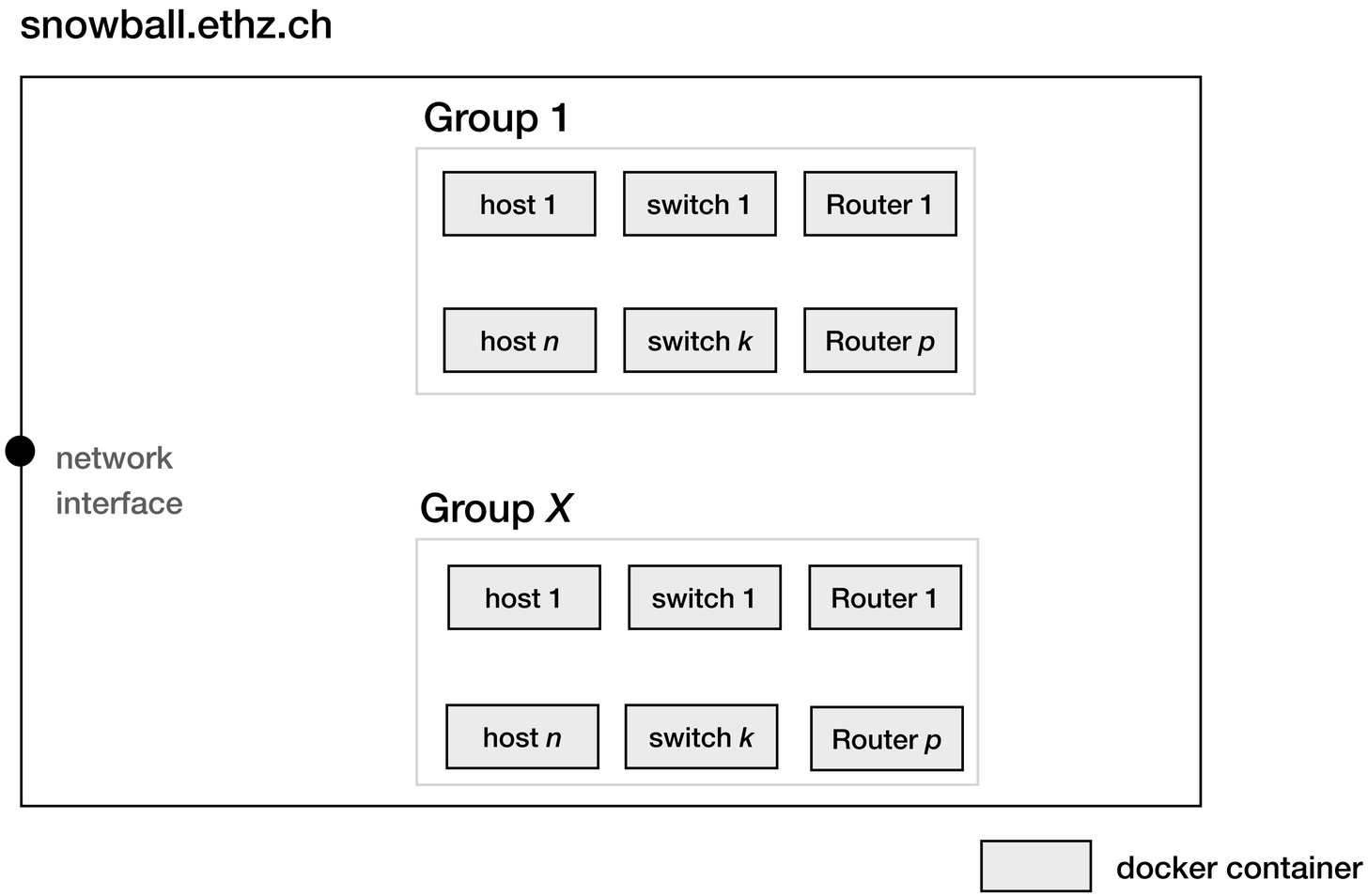
we used VMs between
2016 and 2019

Option #2: linux containers



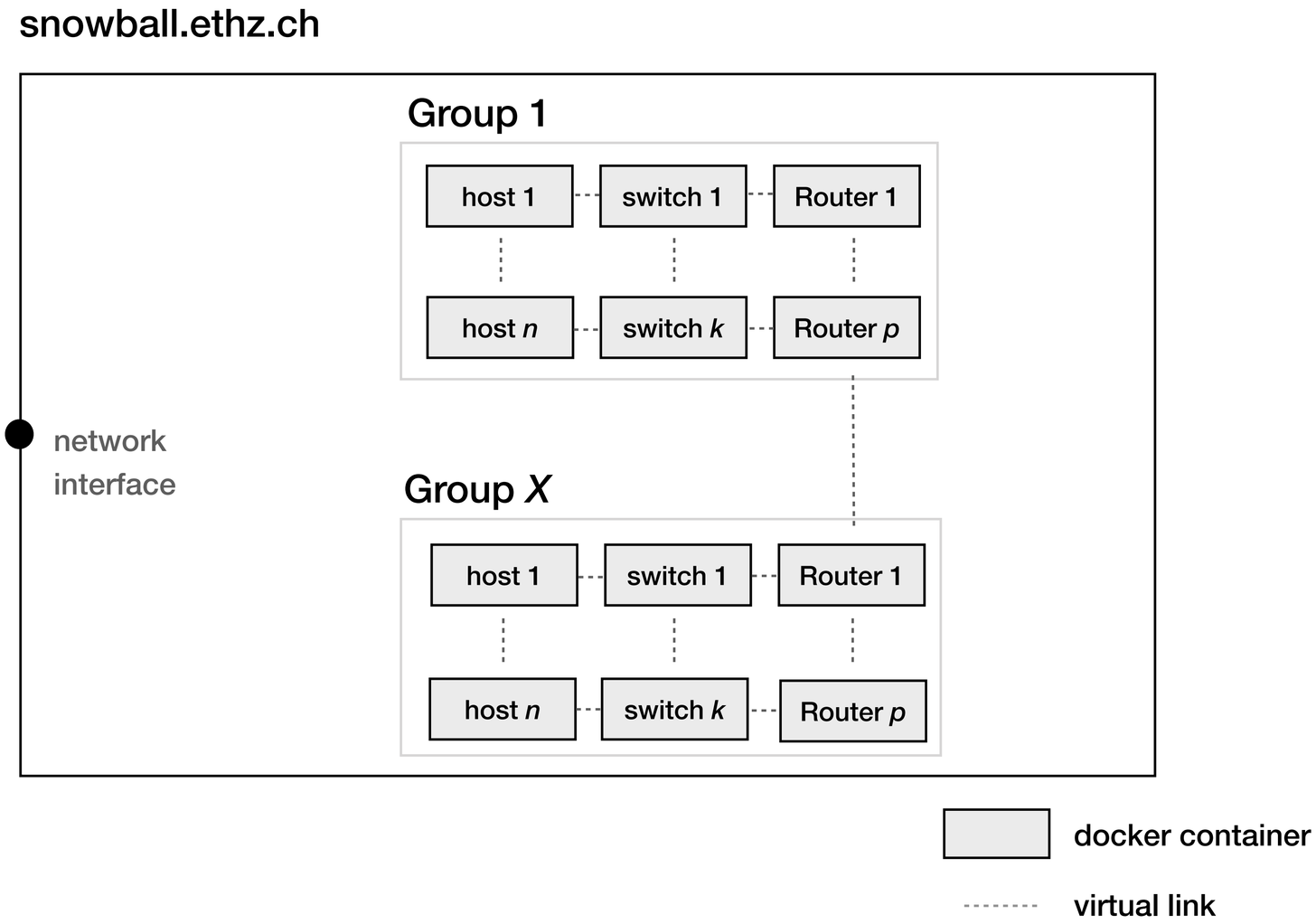
This year

Each router, switch and host runs in its dedicated container

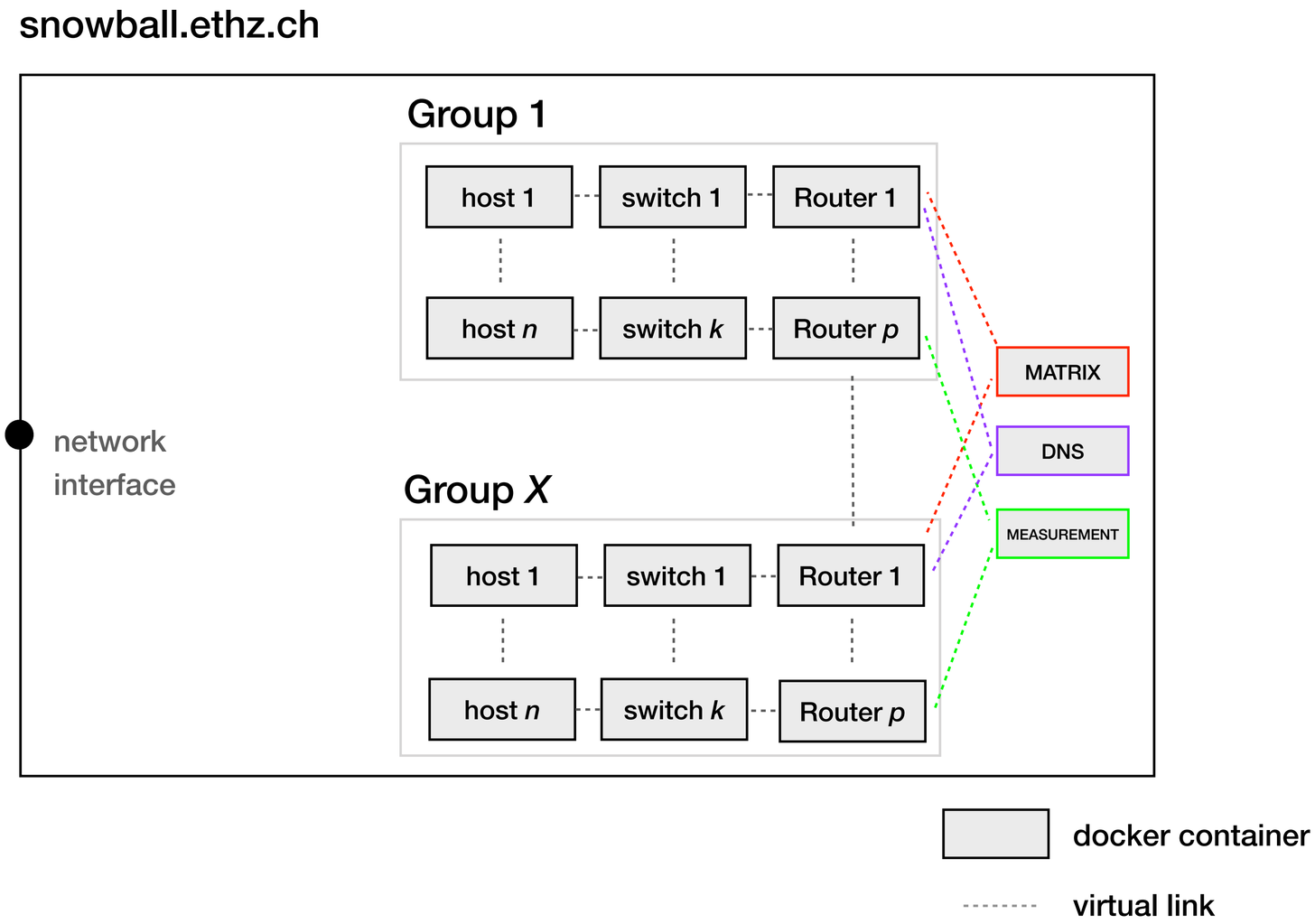


Each router, switch and host runs in its dedicated container

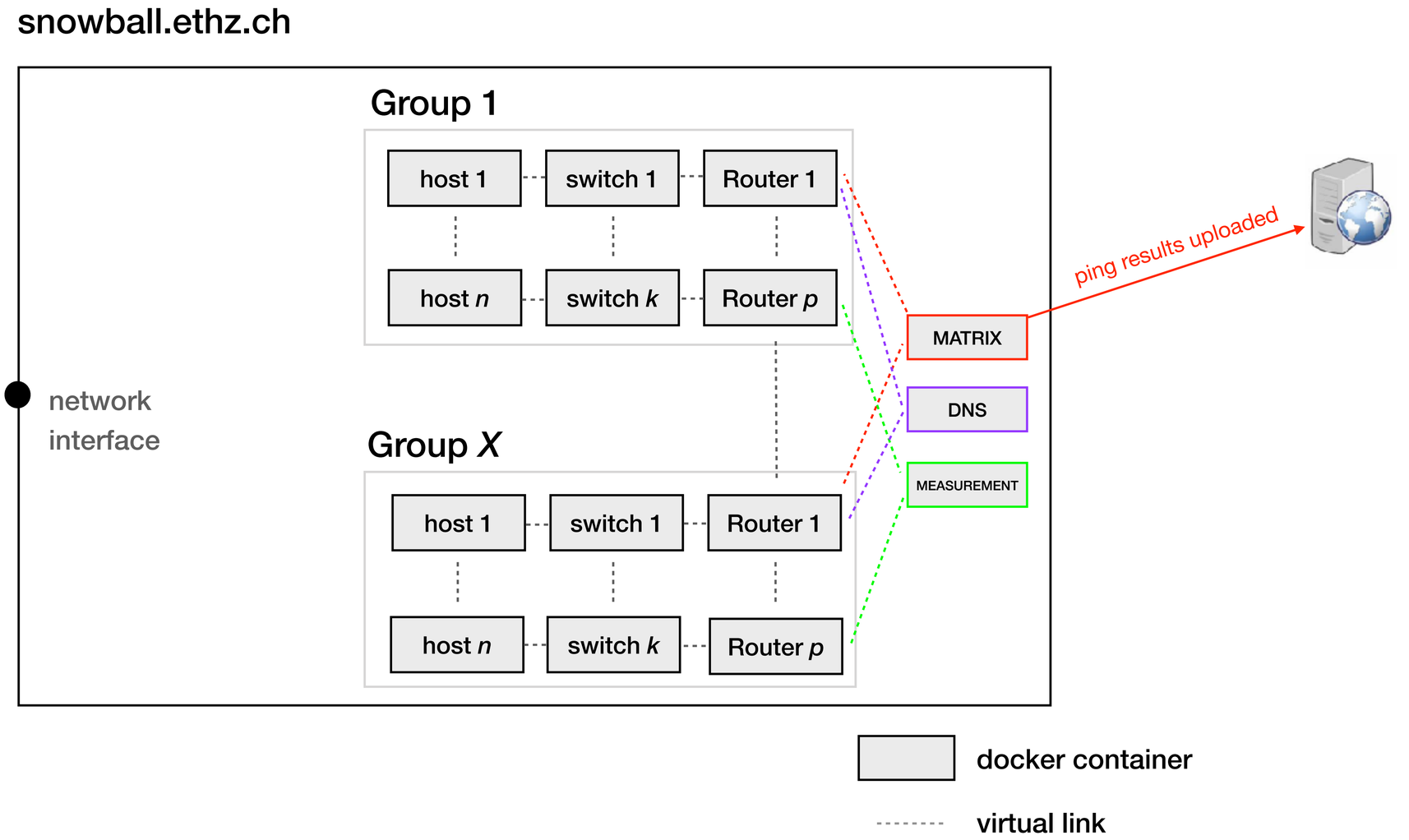
We virtually connect the containers to build the mini-Internet



We use additional containers for the different monitoring services

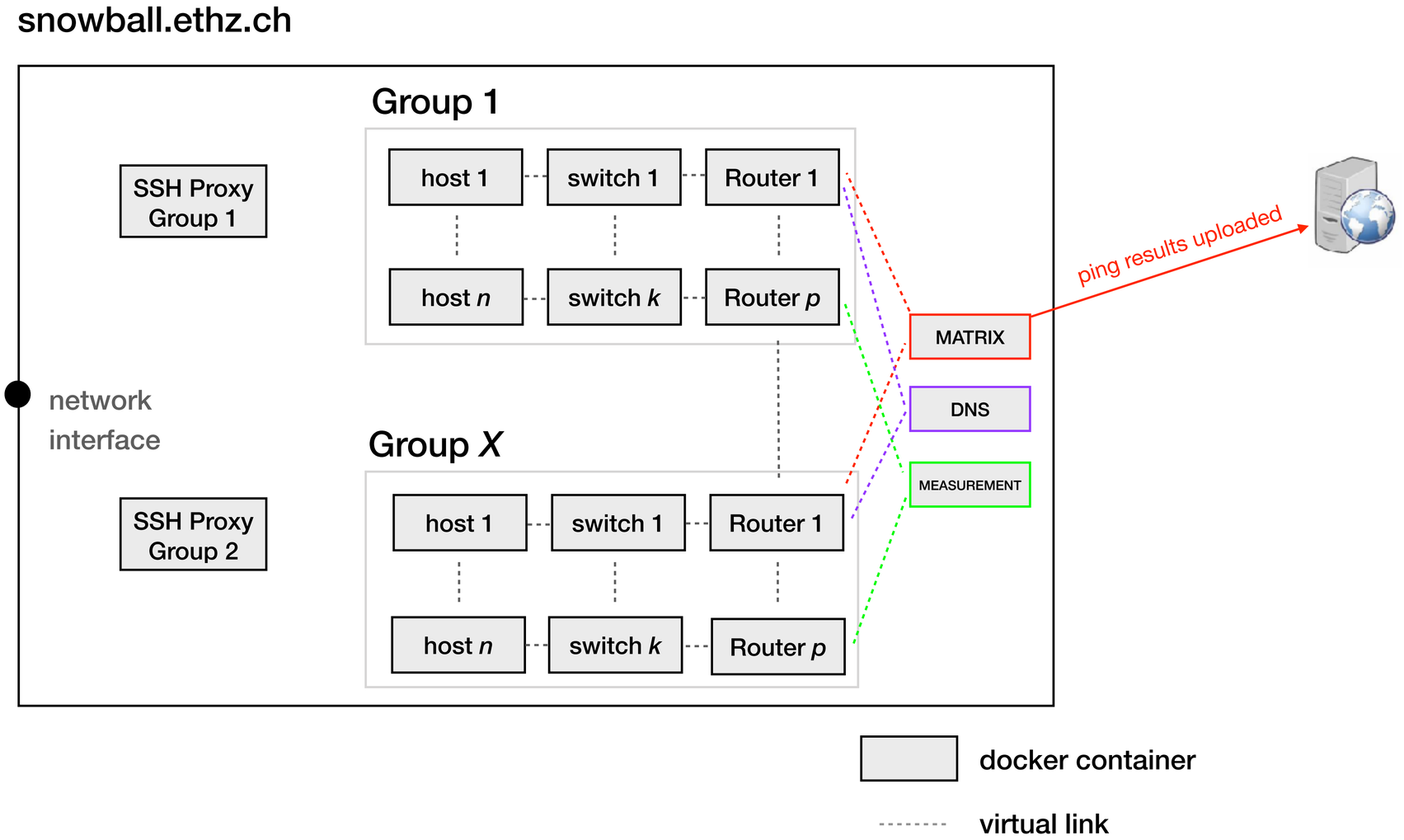


We use additional containers for the different monitoring services

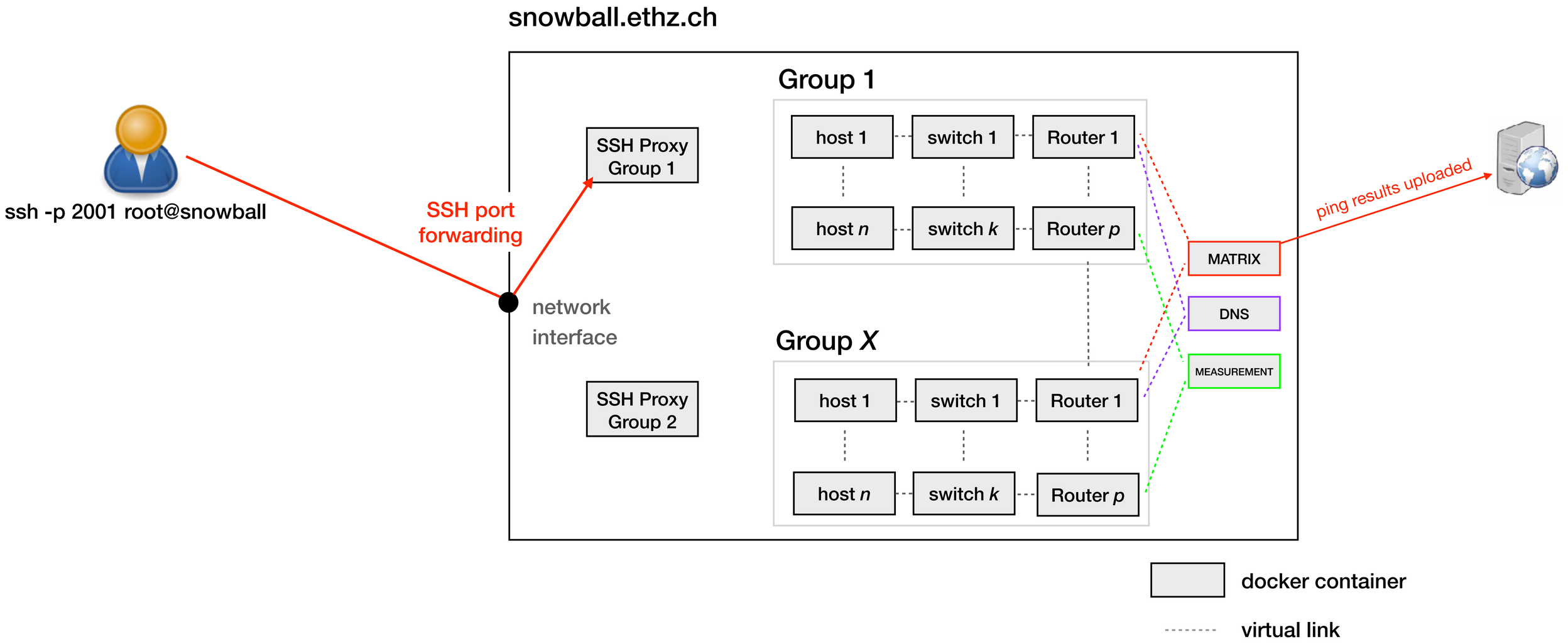


We use "proxy" containers so that you can only access *your* virtual devices

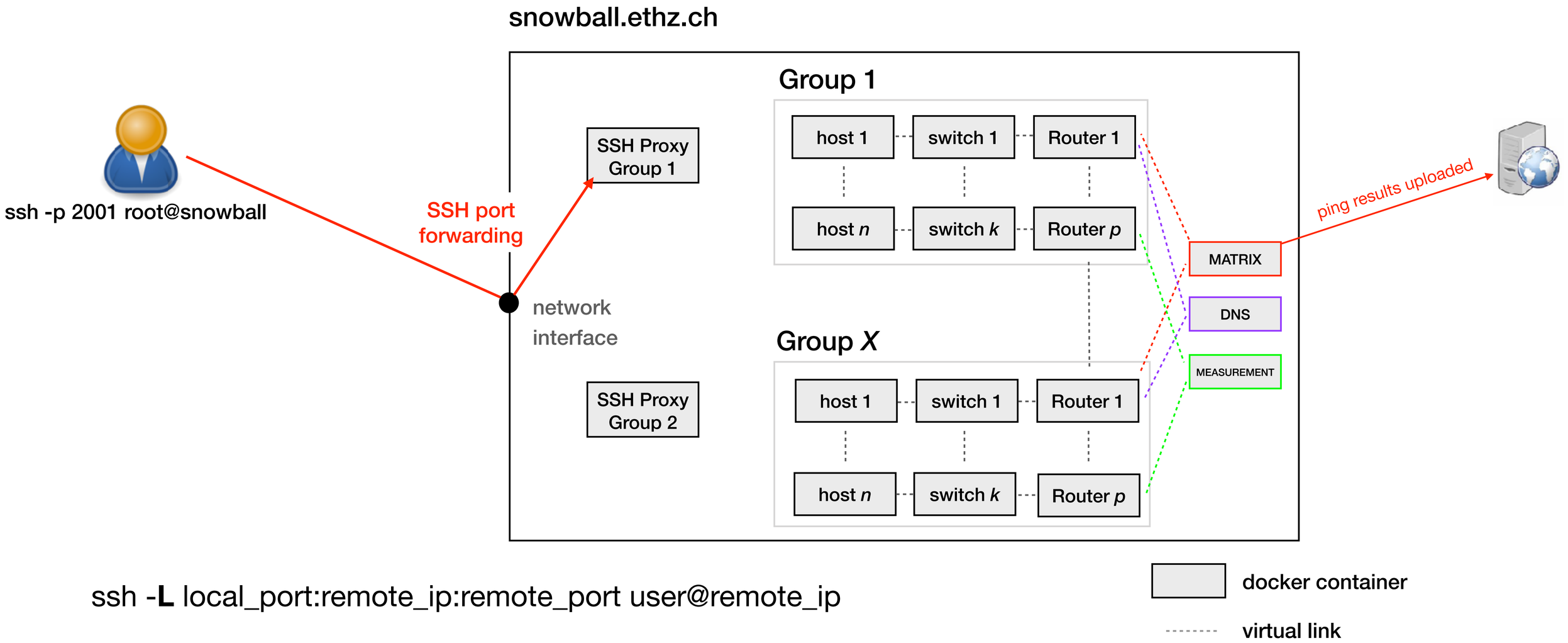
ssh -p 2001 root@snowball



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

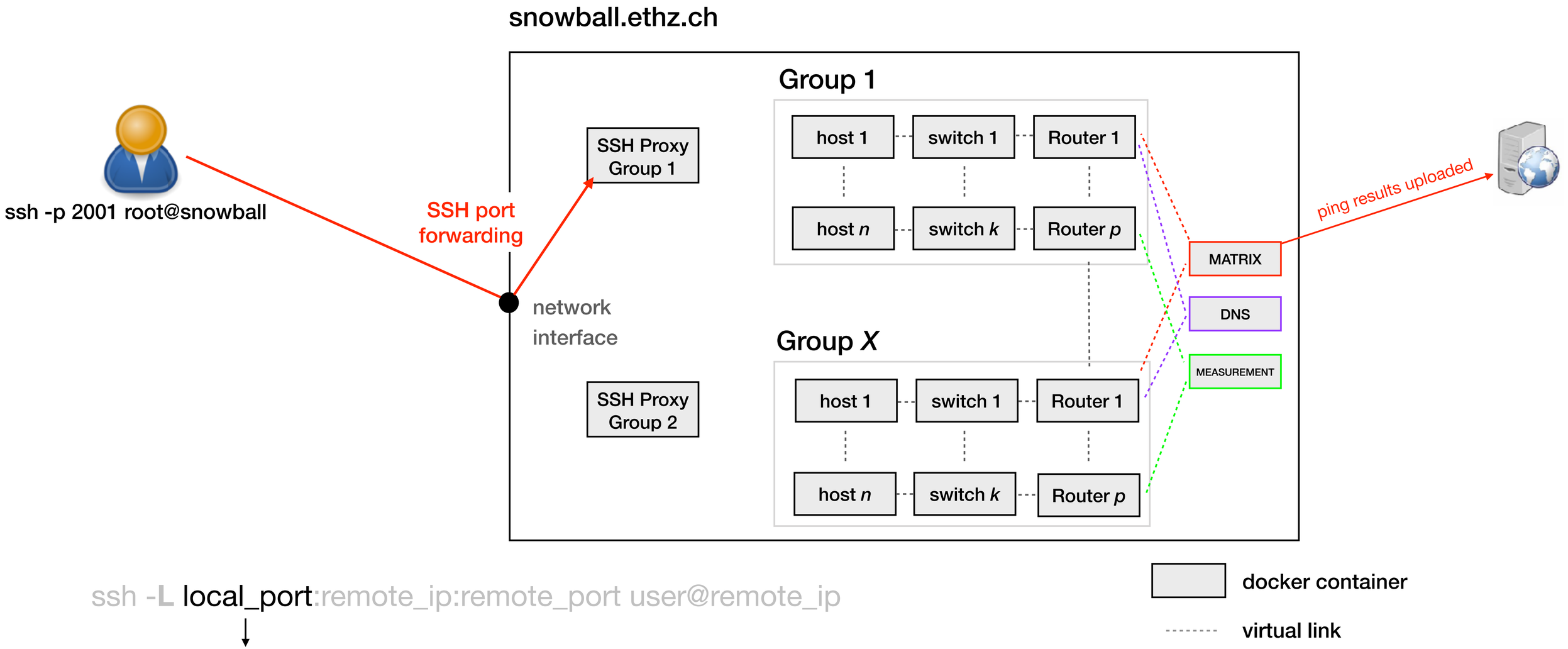


We use "proxy" containers so that you can only access *your* virtual devices



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

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

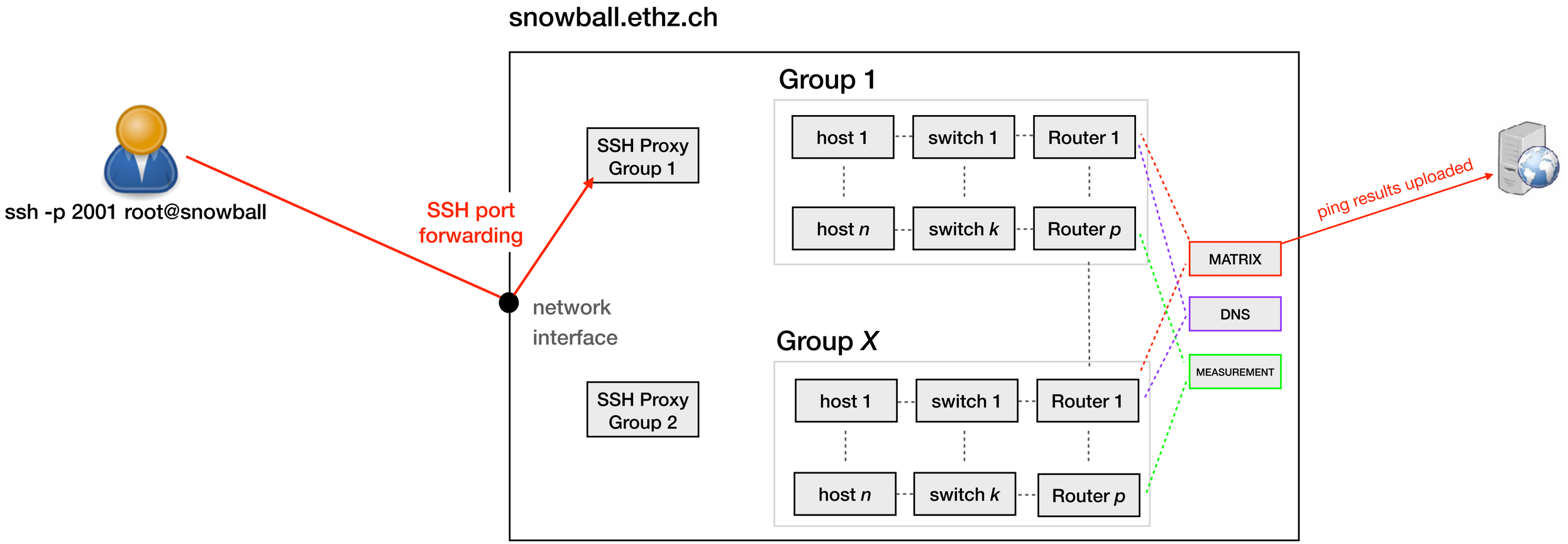


ssh -**L** local_port:remote_ip:remote_port user@remote_ip

↓

2001

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

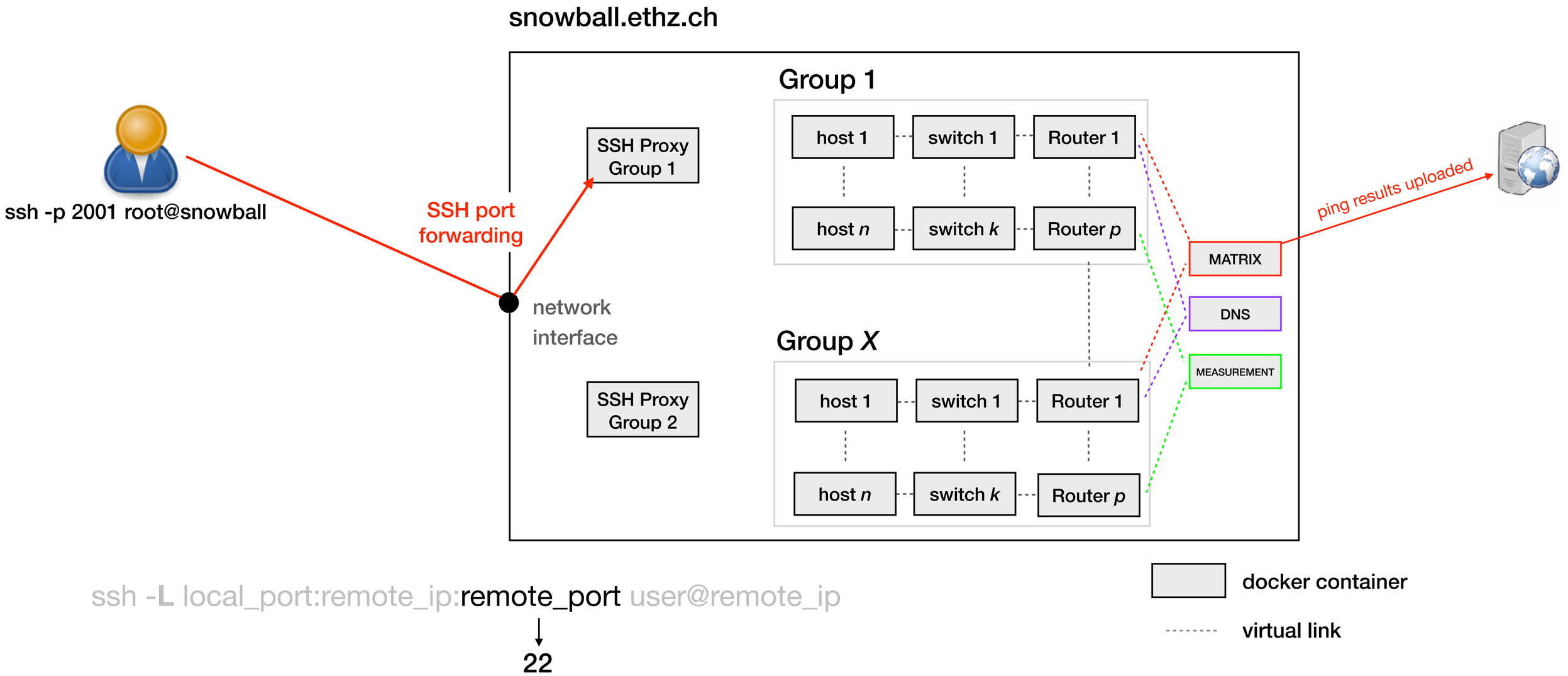


ssh -**L** local_port:remote_ip:remote_port user@remote_ip

↓

IP of SSH Proxy Group1

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

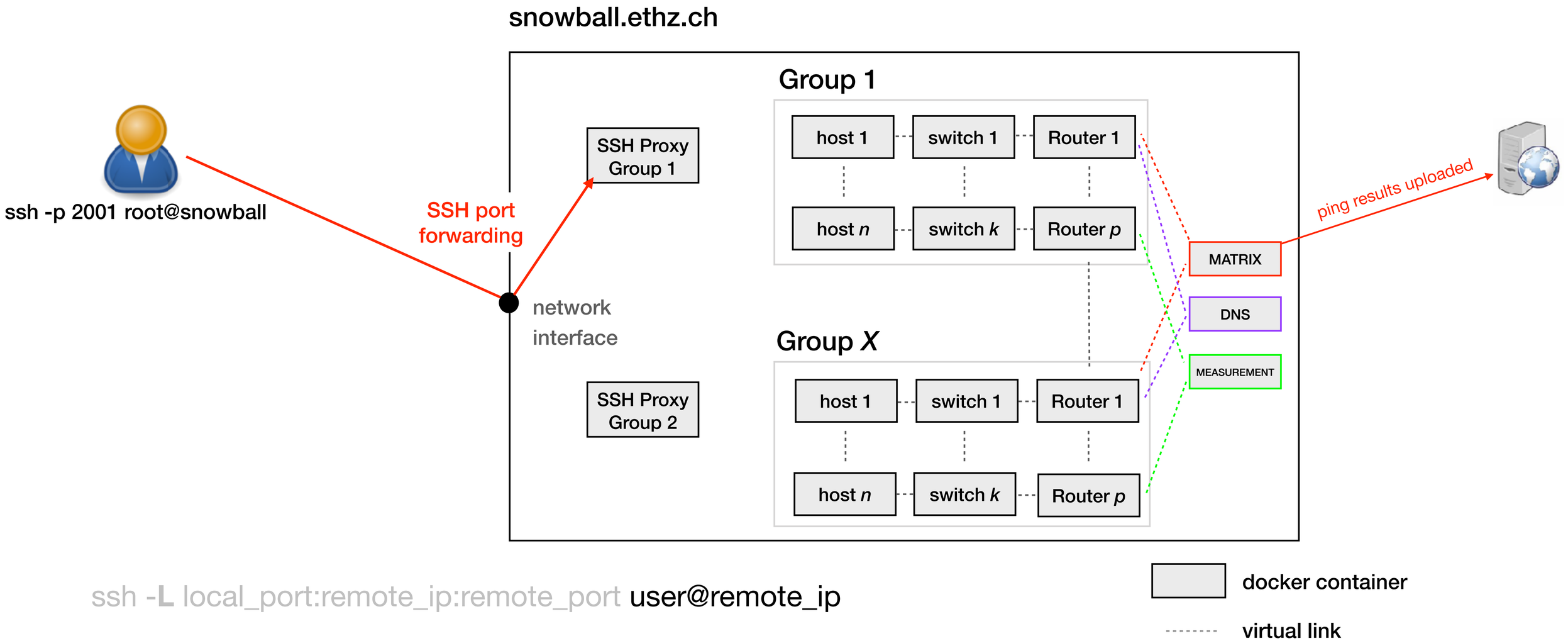


ssh -**L** local_port:remote_ip:remote_port user@remote_ip

↓

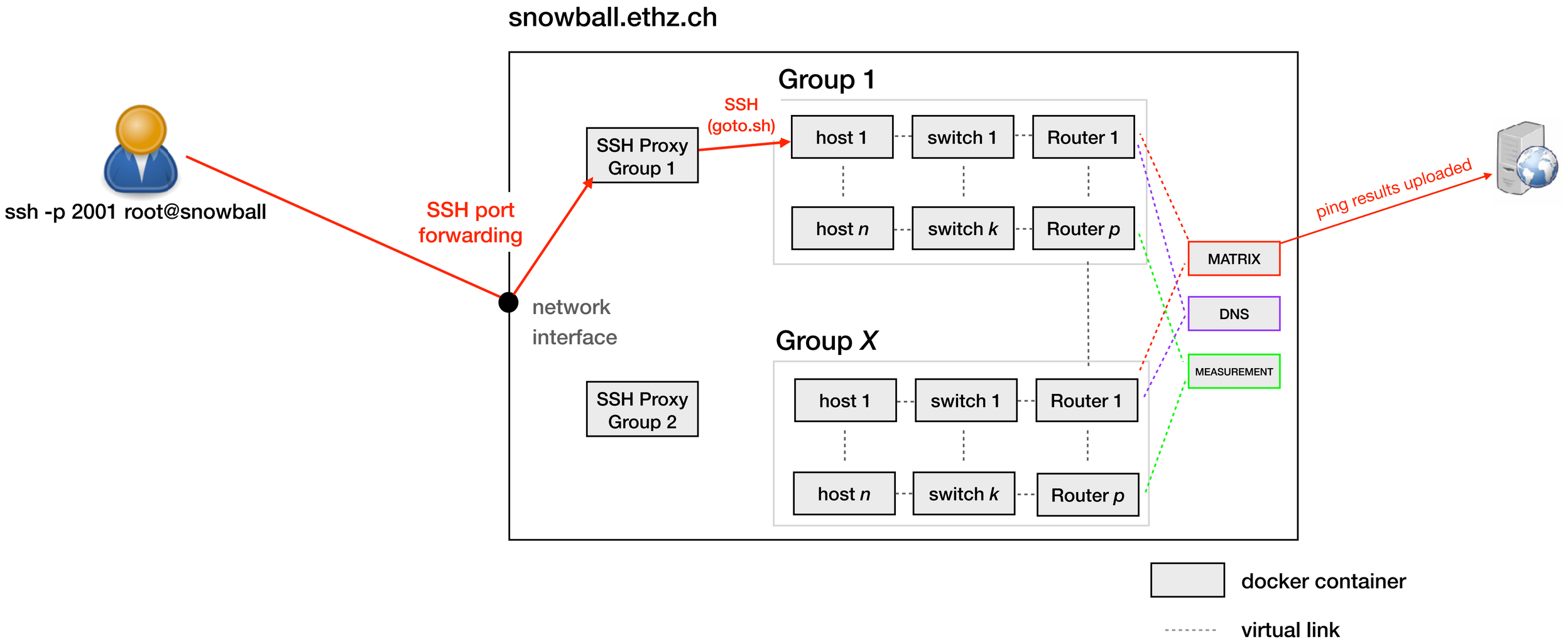
22

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

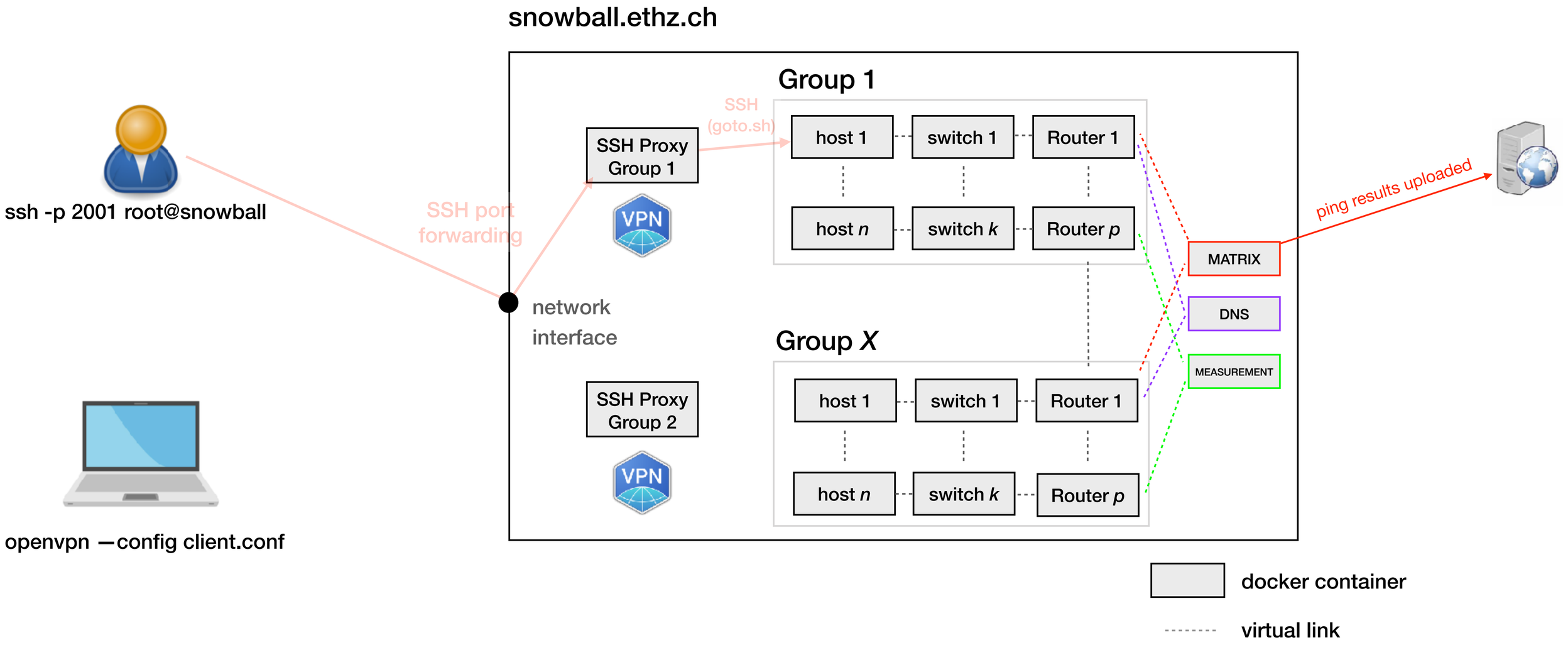


ssh -**L** local_port:remote_ip:remote_port user@remote_ip

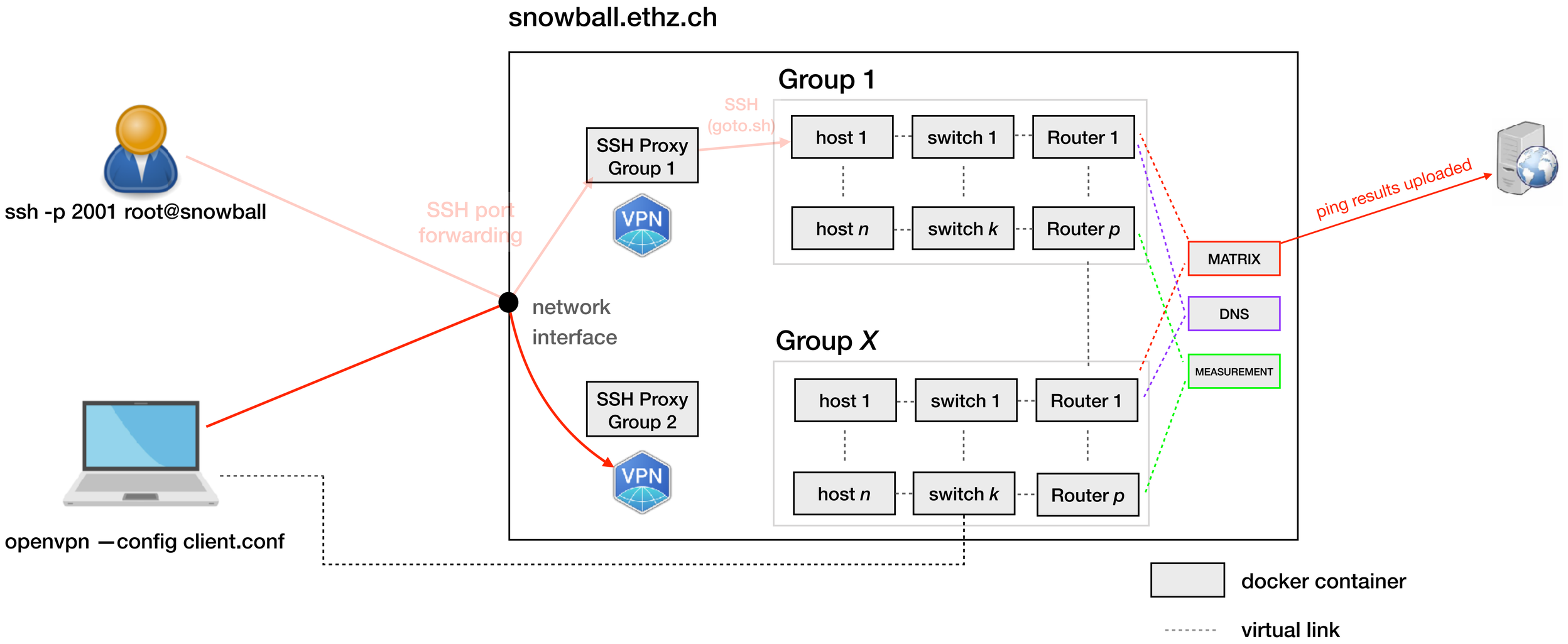
We use "proxy" containers so that you can only access *your* virtual devices



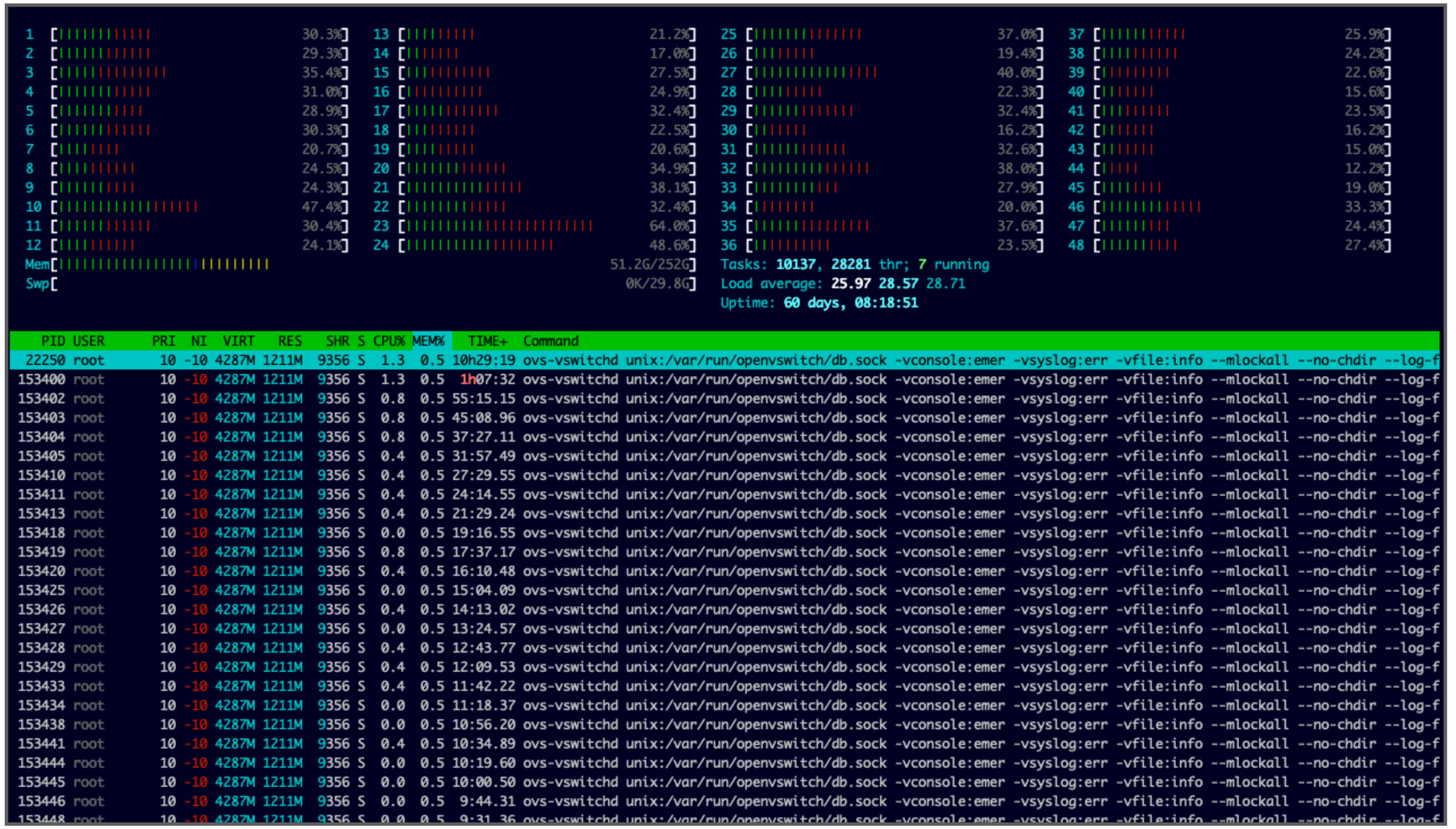
We use "proxy" containers so that you can only access *your* virtual devices



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



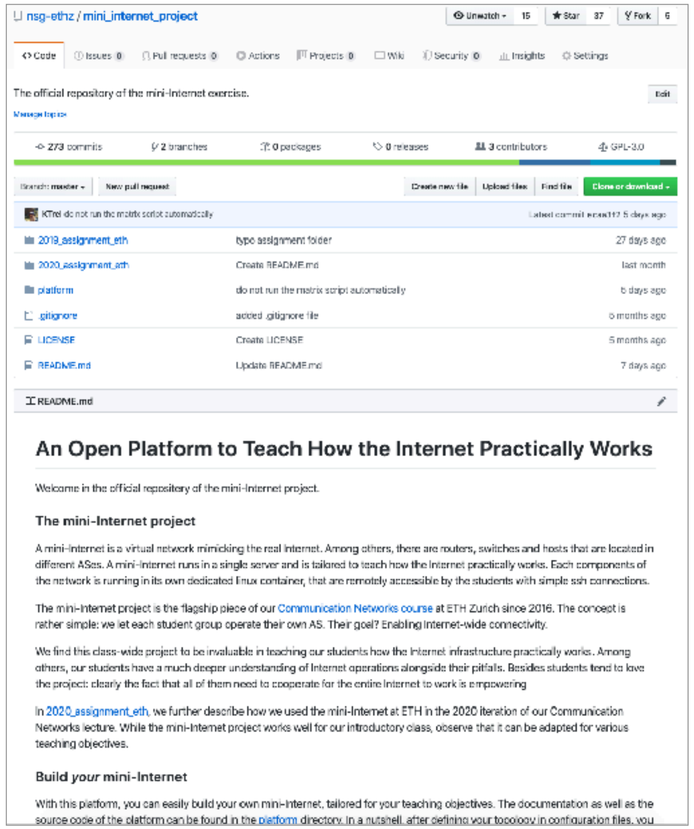
Our server can easily run a 78-ASes mini-Internet



For further information about the mini-Internet: mini-inter.net

Open source implementation

~3700 lines of bash



nsg-ethz / mini_internet_project

Unwatch 15 · Star 37 · Fork 6

Code · Issues 0 · Pull requests 0 · Actions · Projects 0 · Wiki · Security 0 · Insights · Settings

The official repository of the mini-Internet exercise.

Edit

Manage topics

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

Branch: master · New pull request · Create new file · Upload files · Find file · Clone or download

KTrel do not run the matrix script automatically · Latest commit a1ca117 5 days ago

2019_assignment_eth	typo 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

README.md

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.

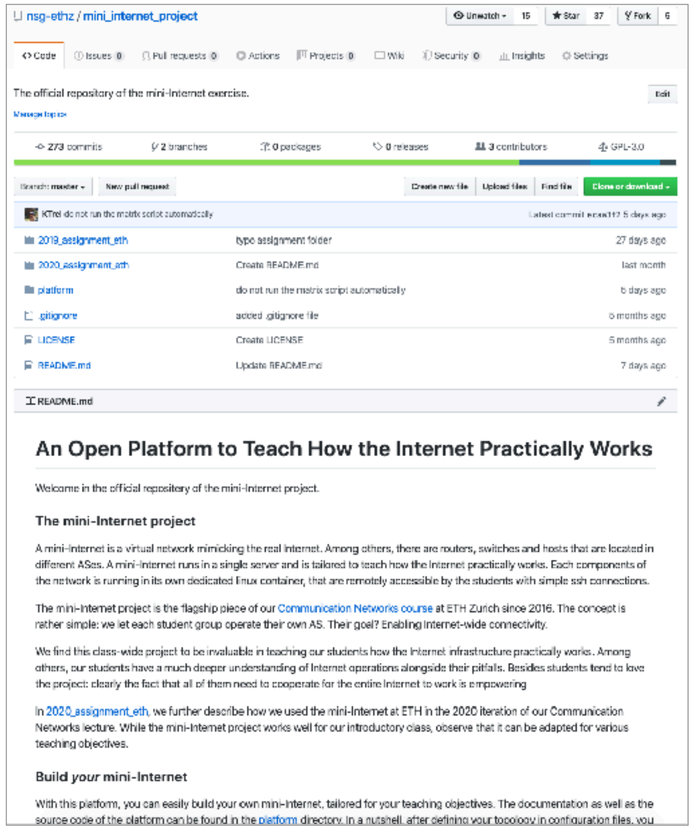
In 2020_assignment_eth, we further describe how we used the mini-Internet at ETH in the 2020 iteration of our Communication Networks lecture. While the mini-Internet project works well for our introductory class, observe that it can be adapted for various teaching objectives.

Build *your* mini-Internet

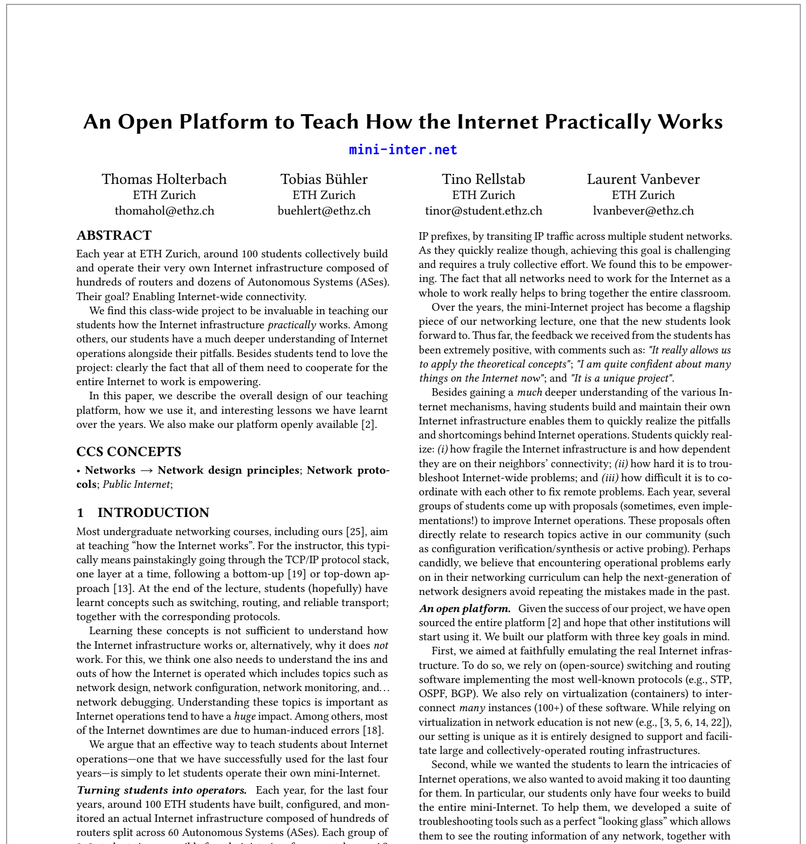
With this platform, you can easily build your own mini-Internet, tailored for your teaching objectives. The documentation as well as the source code of the platform can be found in the platform directory. In a nutshell, after defining your topology in configuration files, you

For further information about the mini-Internet: mini-inter.net

Open source implementation
~3700 lines of bash



Published in SIGCOMM CCR’20



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 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.

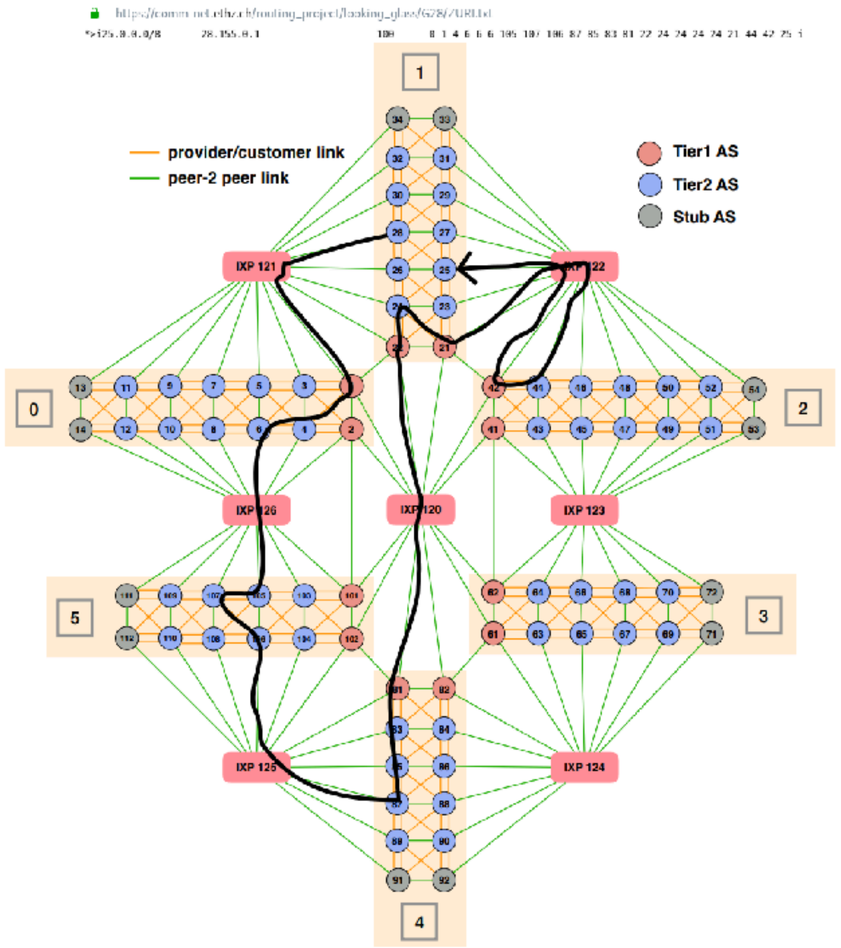
First, we aimed at faithfully emulating the real Internet infrastructure. To do so, we rely on (open-source) switching and routing software implementing the most well-known protocols (e.g., STP, OSPF, BGP). We also rely on virtualization (containers) to interconnect *many* instances (100+) of these software.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

Youtube video

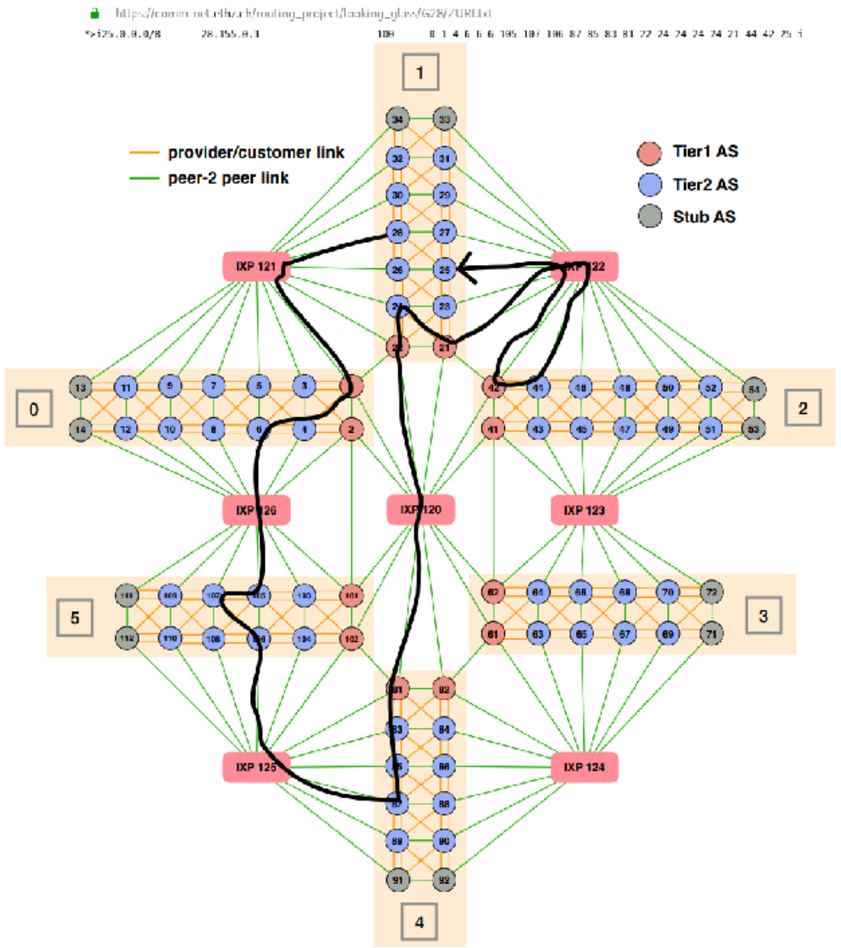
Every year we improve the project

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

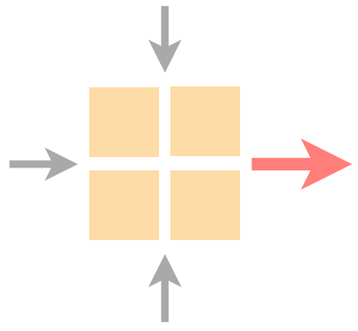
We plan to implement the RPKI infrastructure into the mini-internet so that you can validate the origin of the BGP routes

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

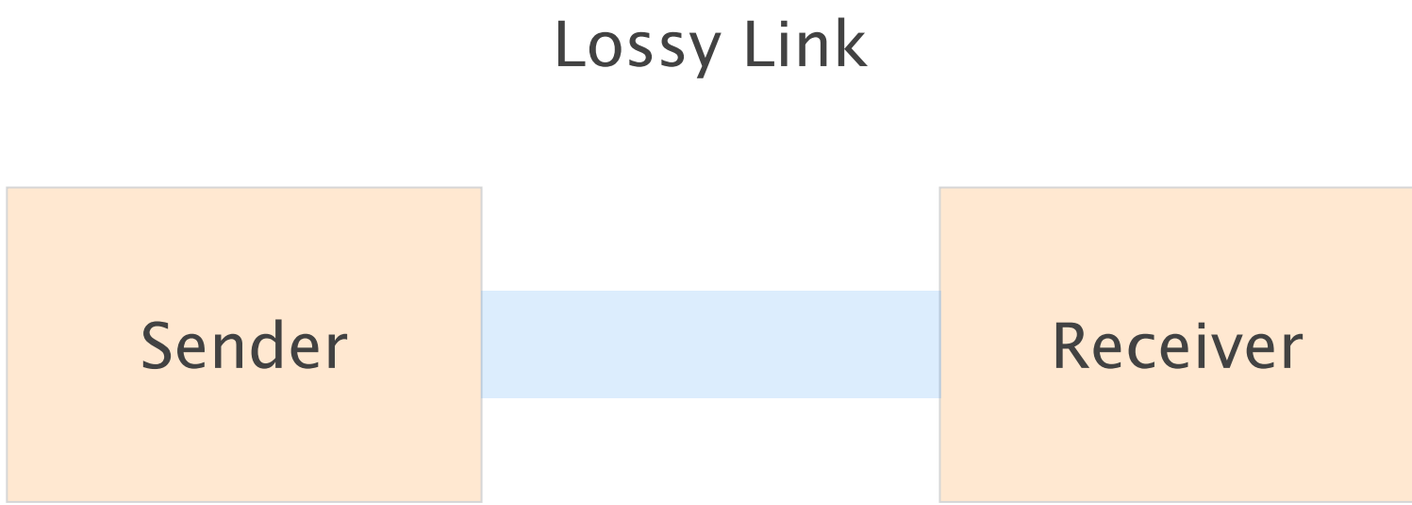
Current assignment

Implement your own Reliable Transport Protocol

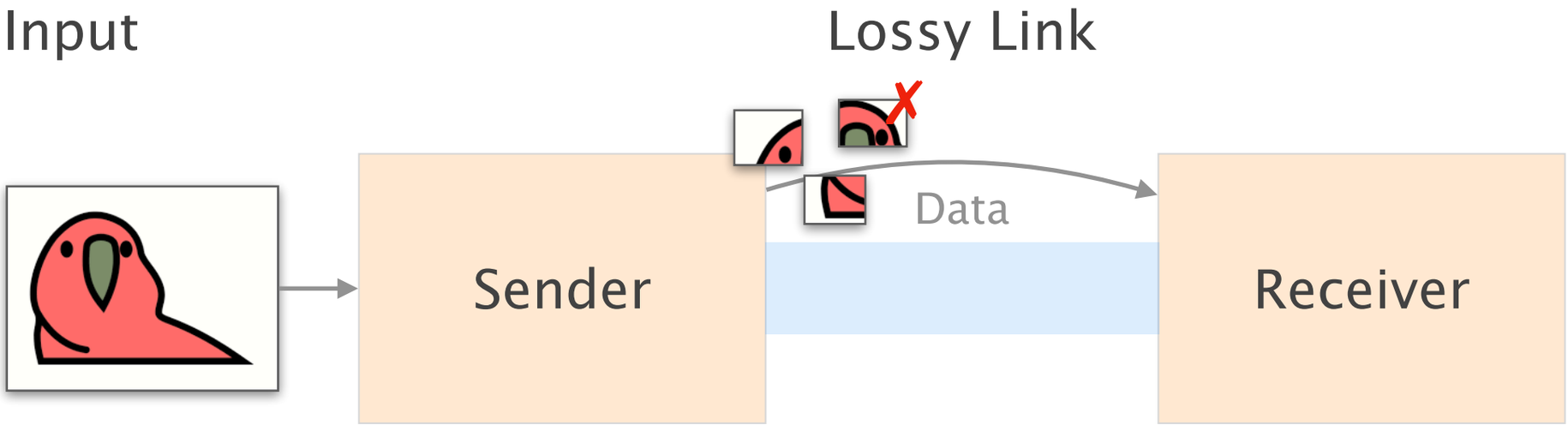
Implement your own Reliable Transport Protocol

recover from packet loss
and reordering

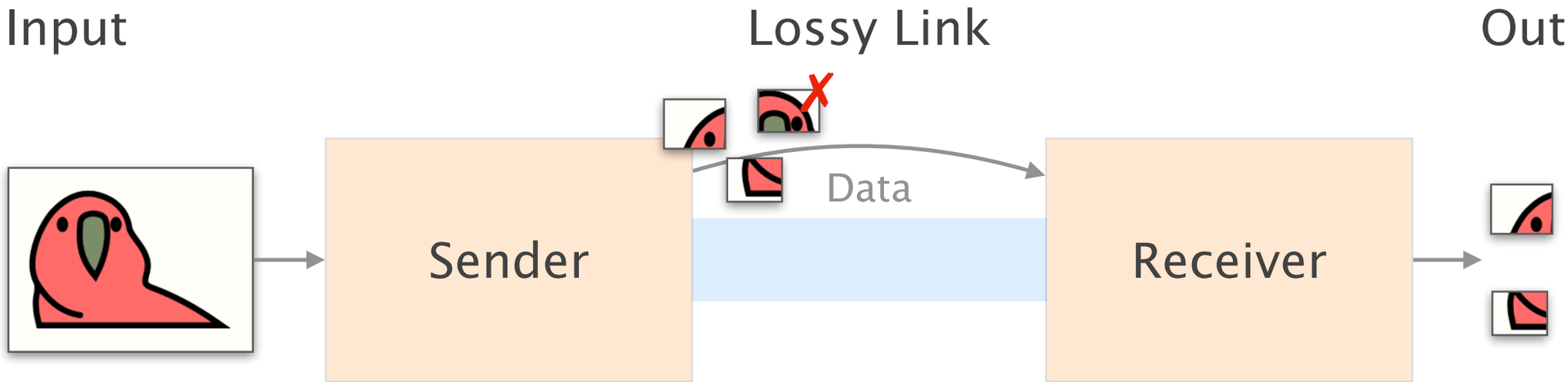
Implement your own Reliable Transport Protocol



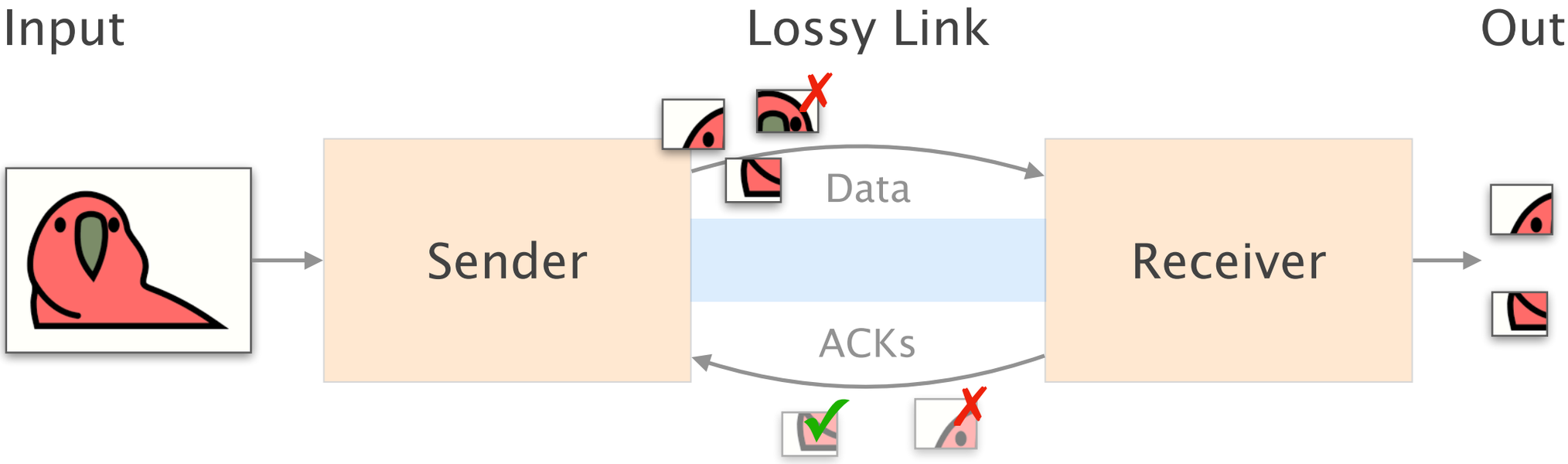
Implement your own Reliable Transport Protocol



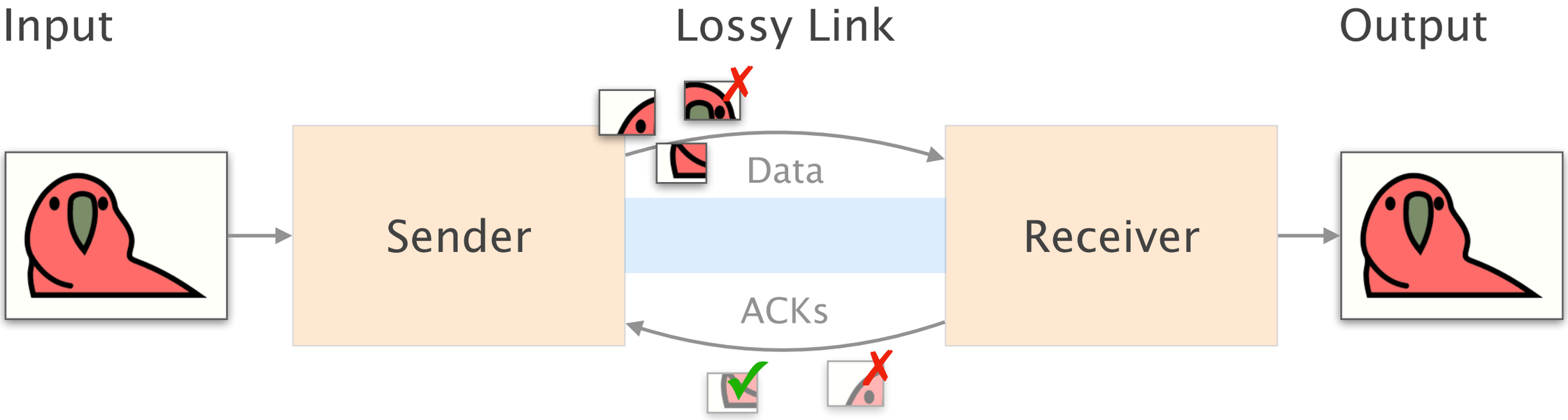
Implement your own Reliable Transport Protocol



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

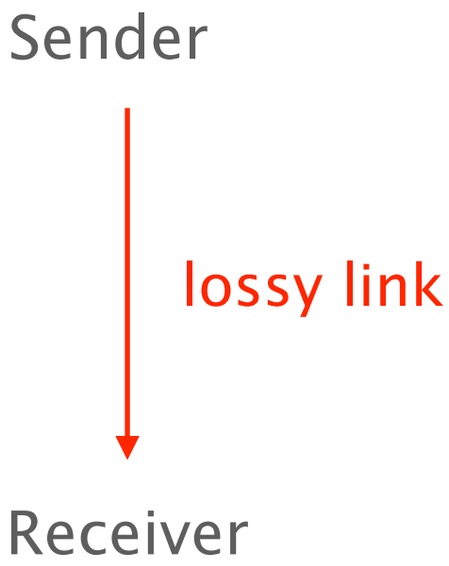
Sender

Receiver

The Go-Back-N Protocol

a simple reliable transport protocol with

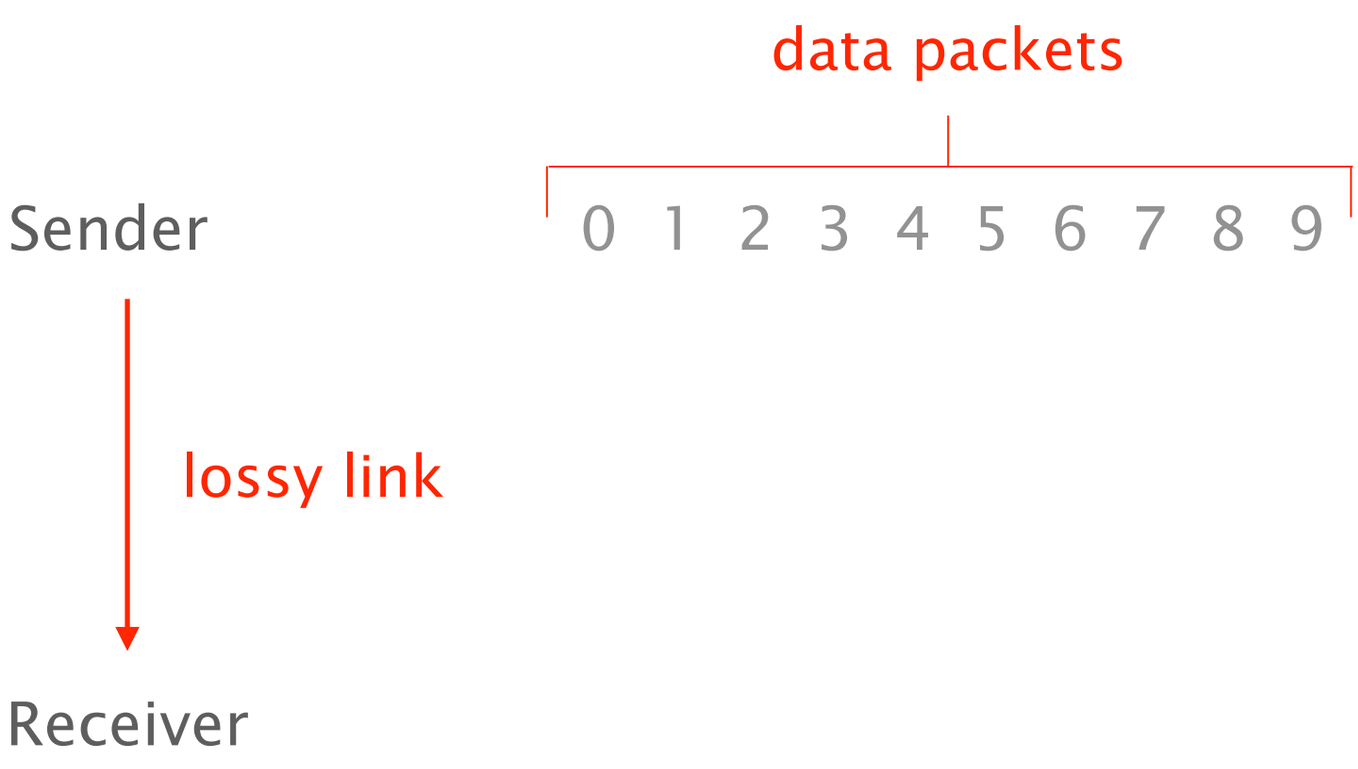
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

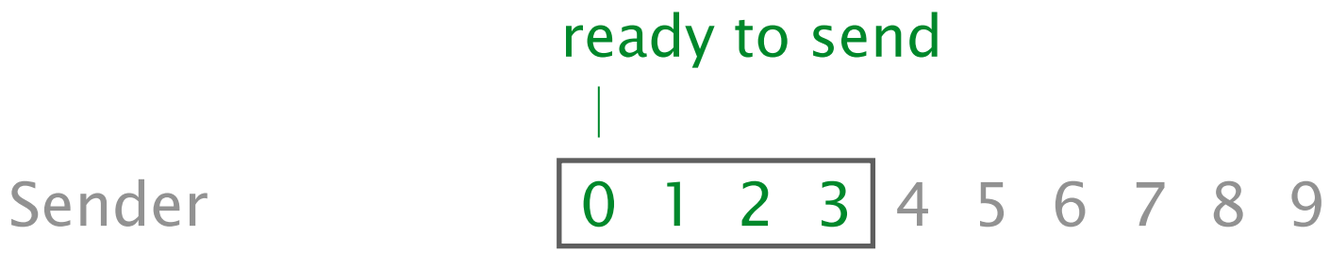
a simple reliable transport protocol with

a sliding window, cumulative ACKs, timeouts and retransmissions

The Go-Back-N Protocol

a simple reliable transport protocol with

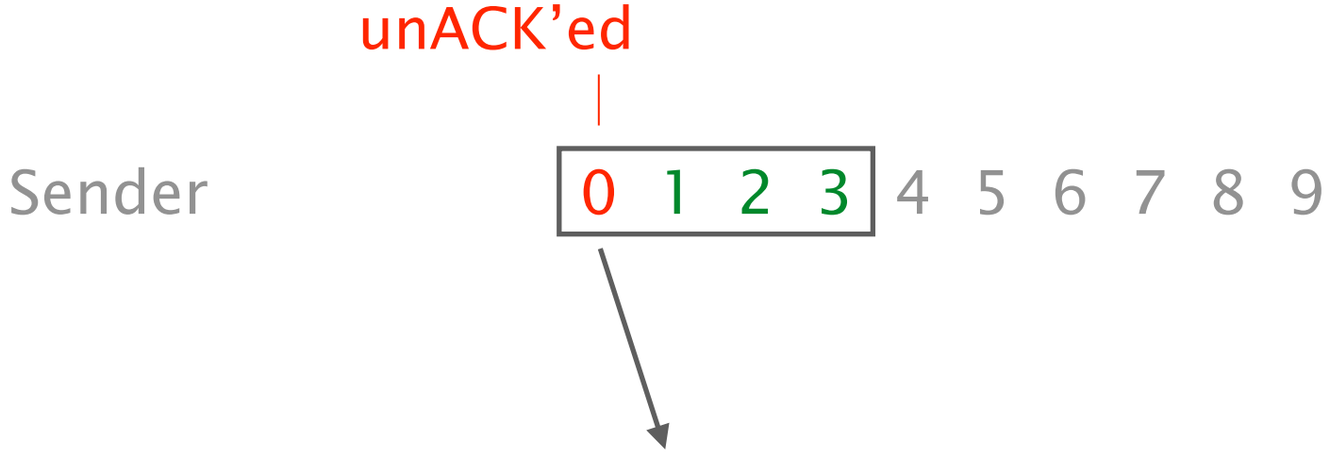
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

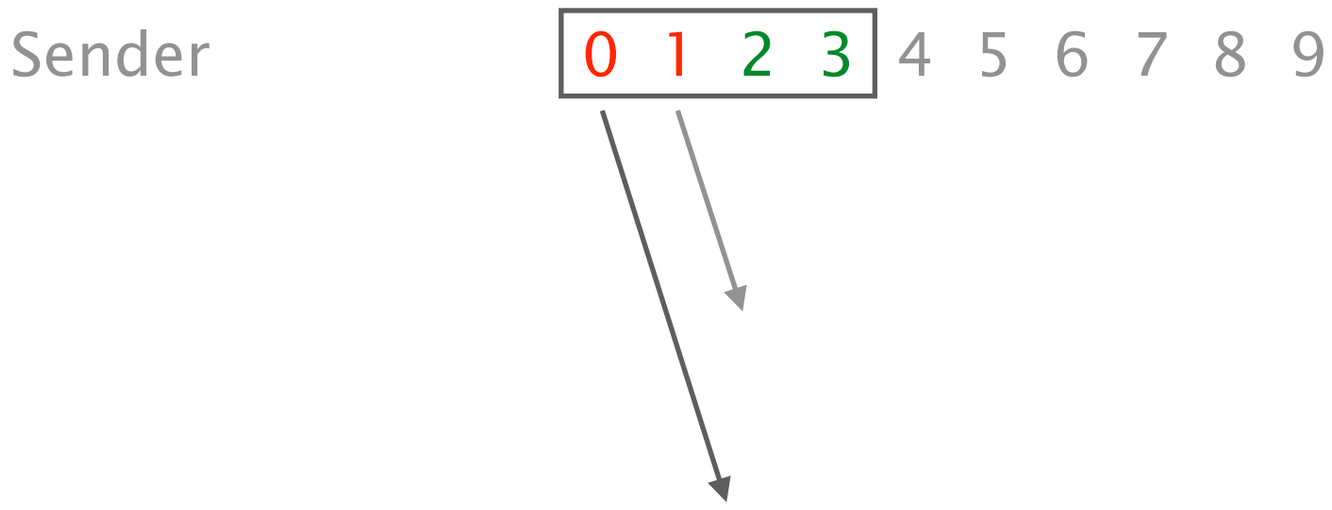
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

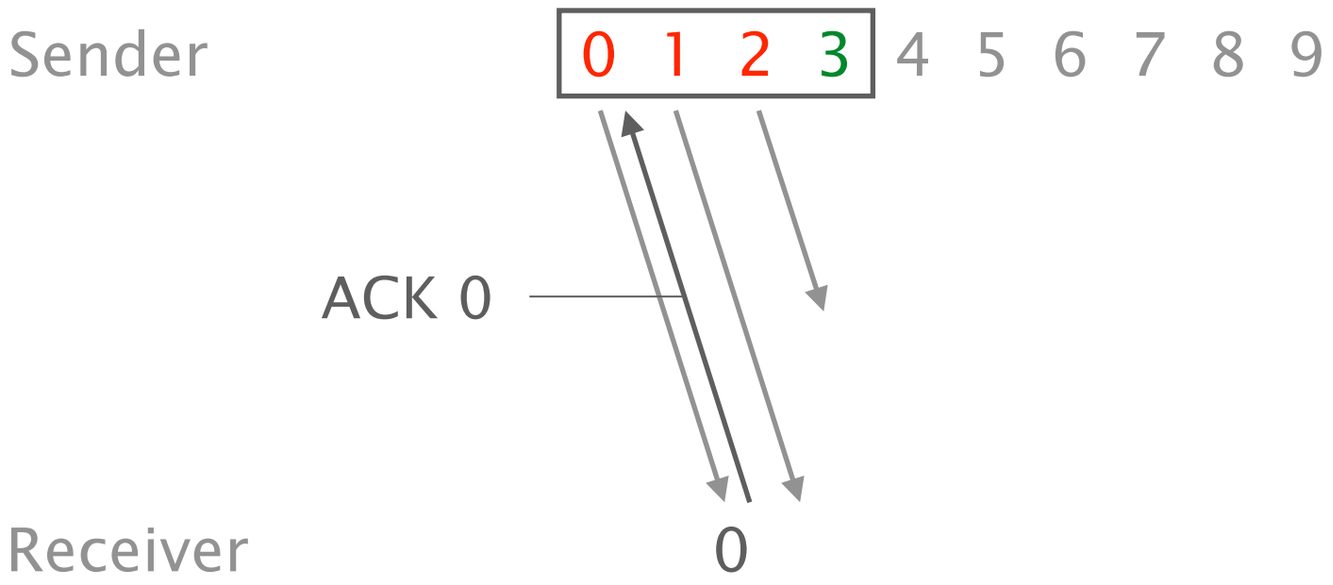
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

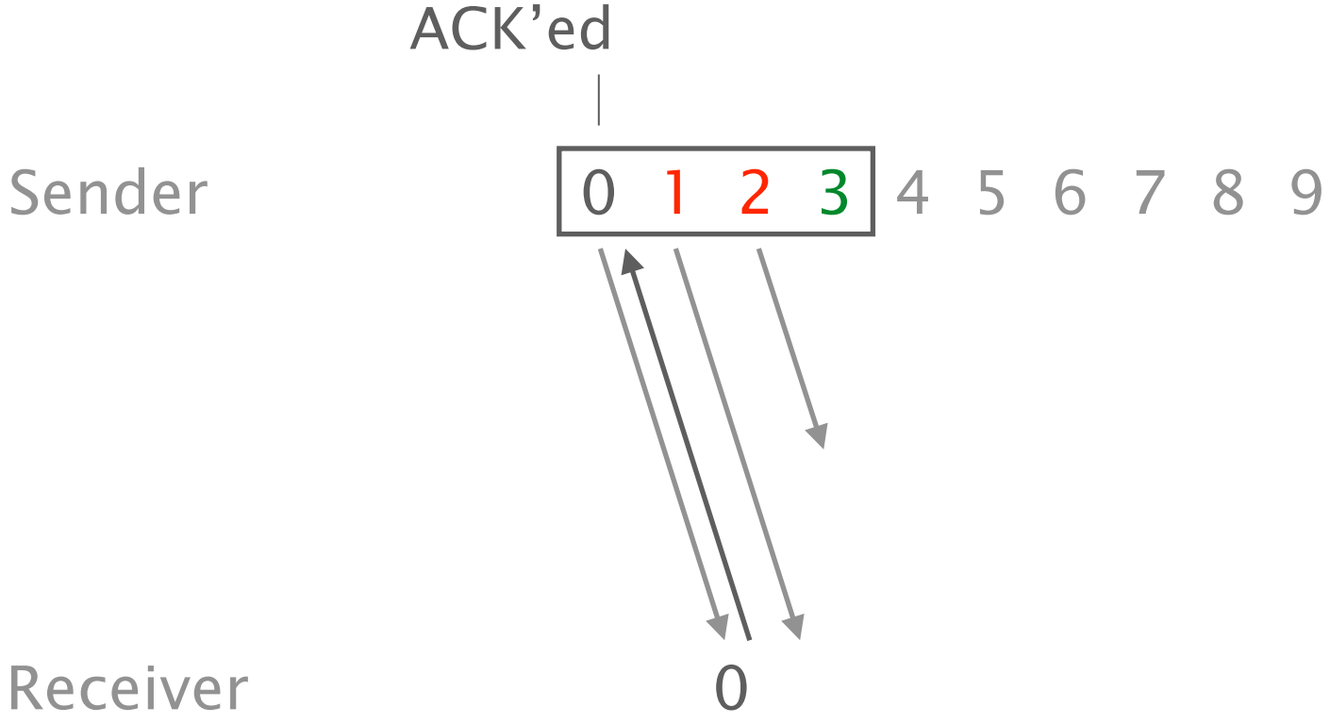
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

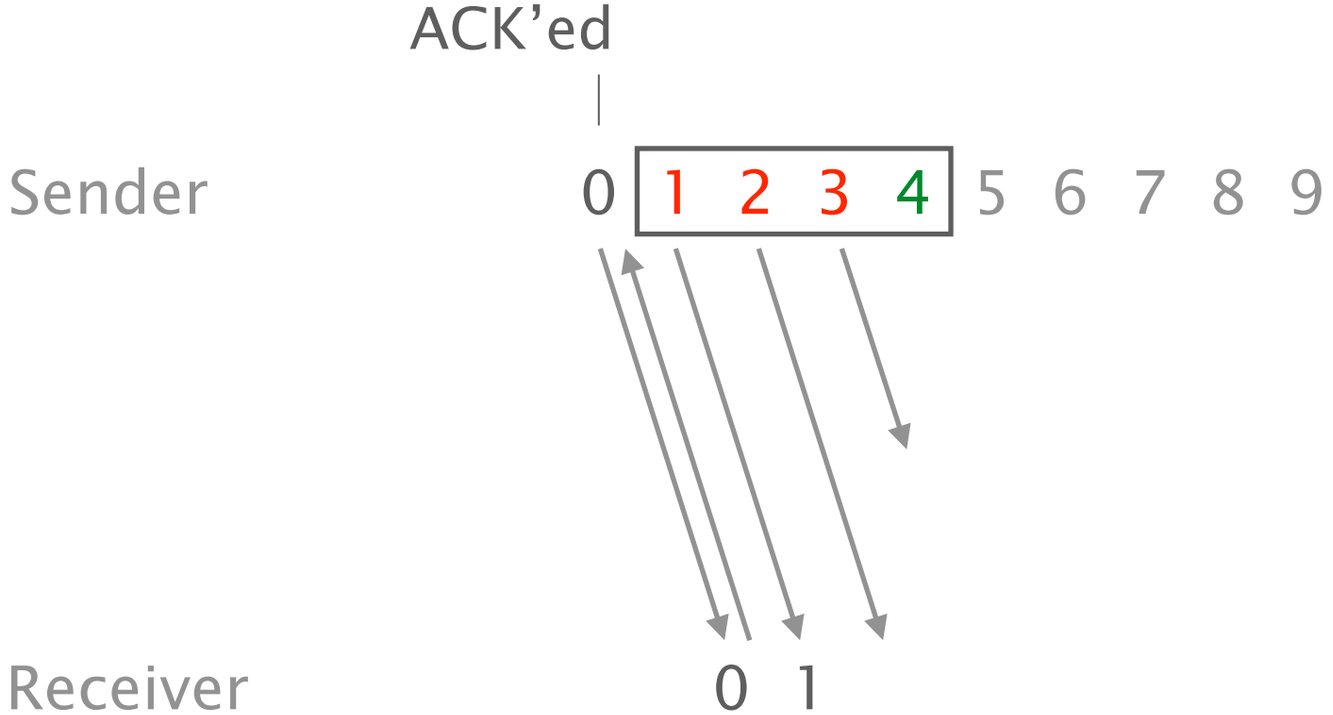
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

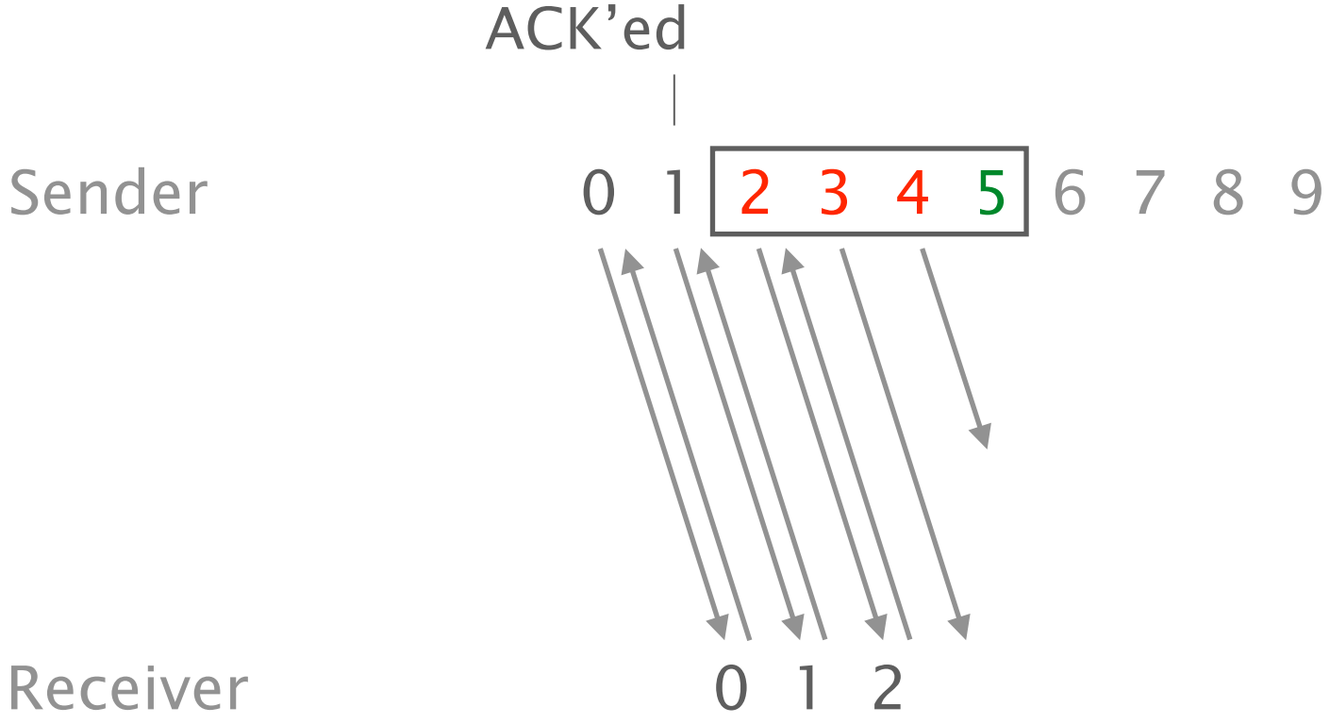
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

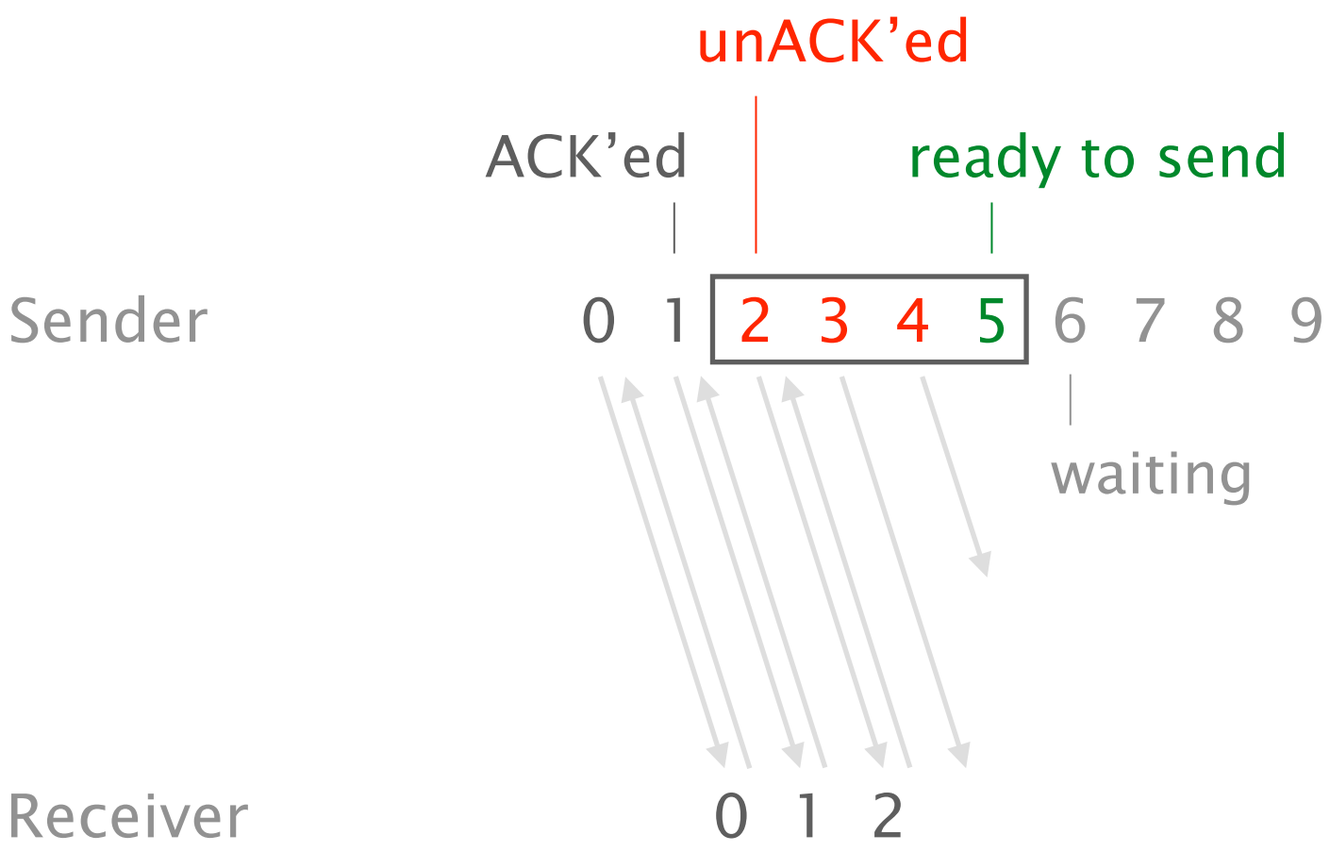
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

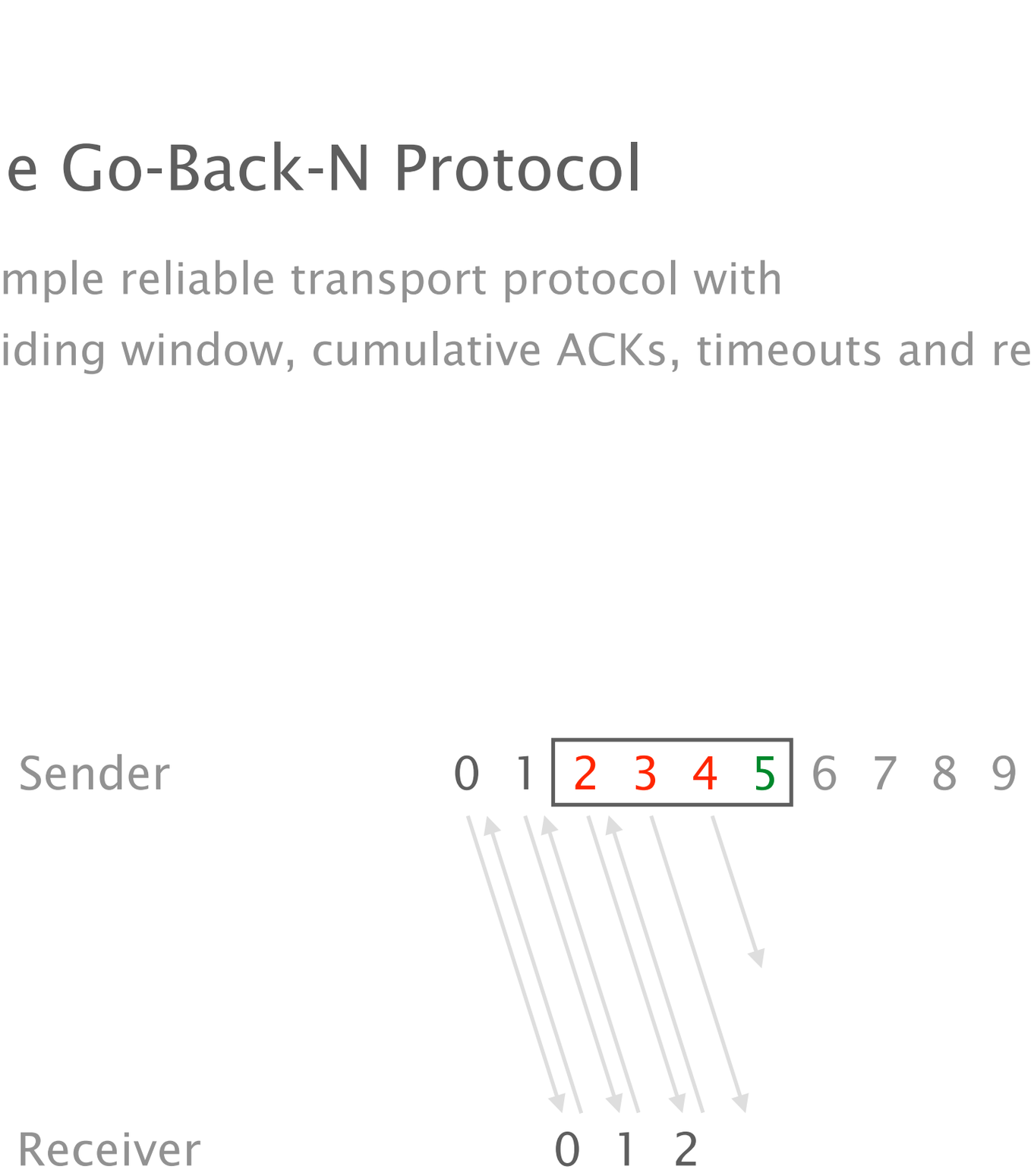
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

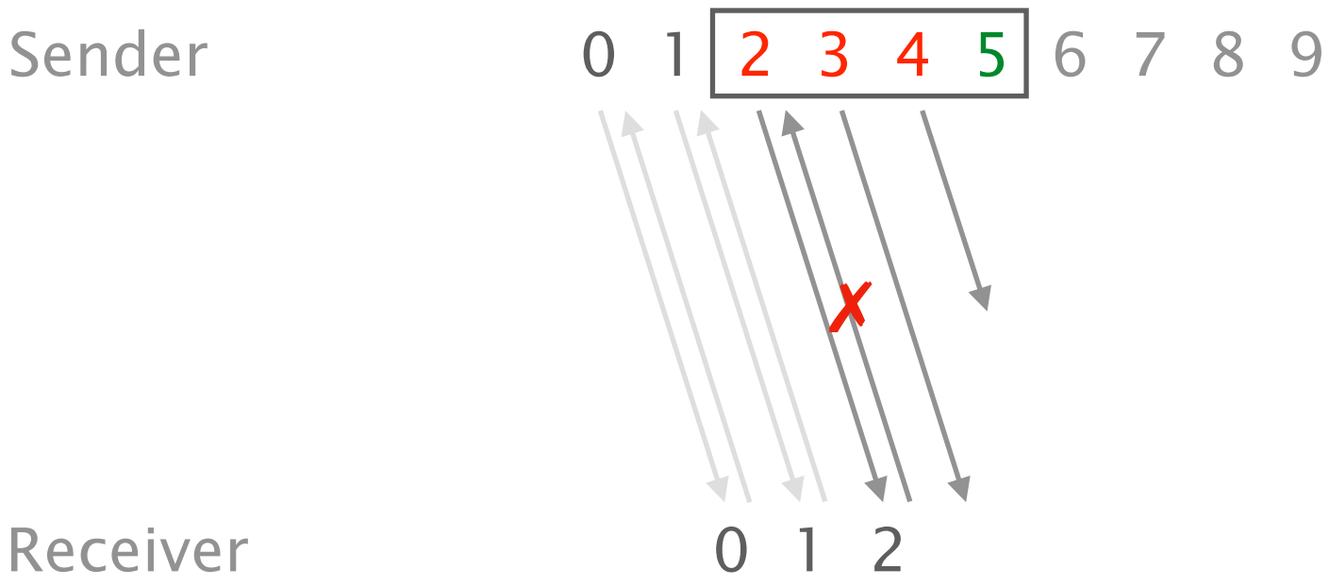
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

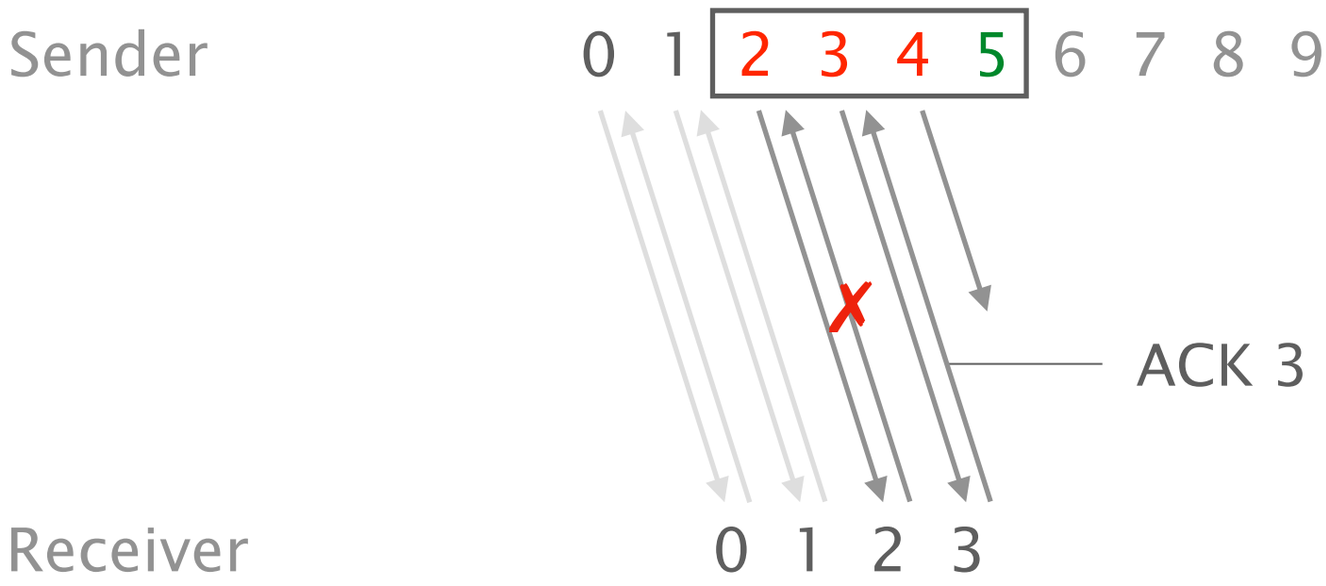
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

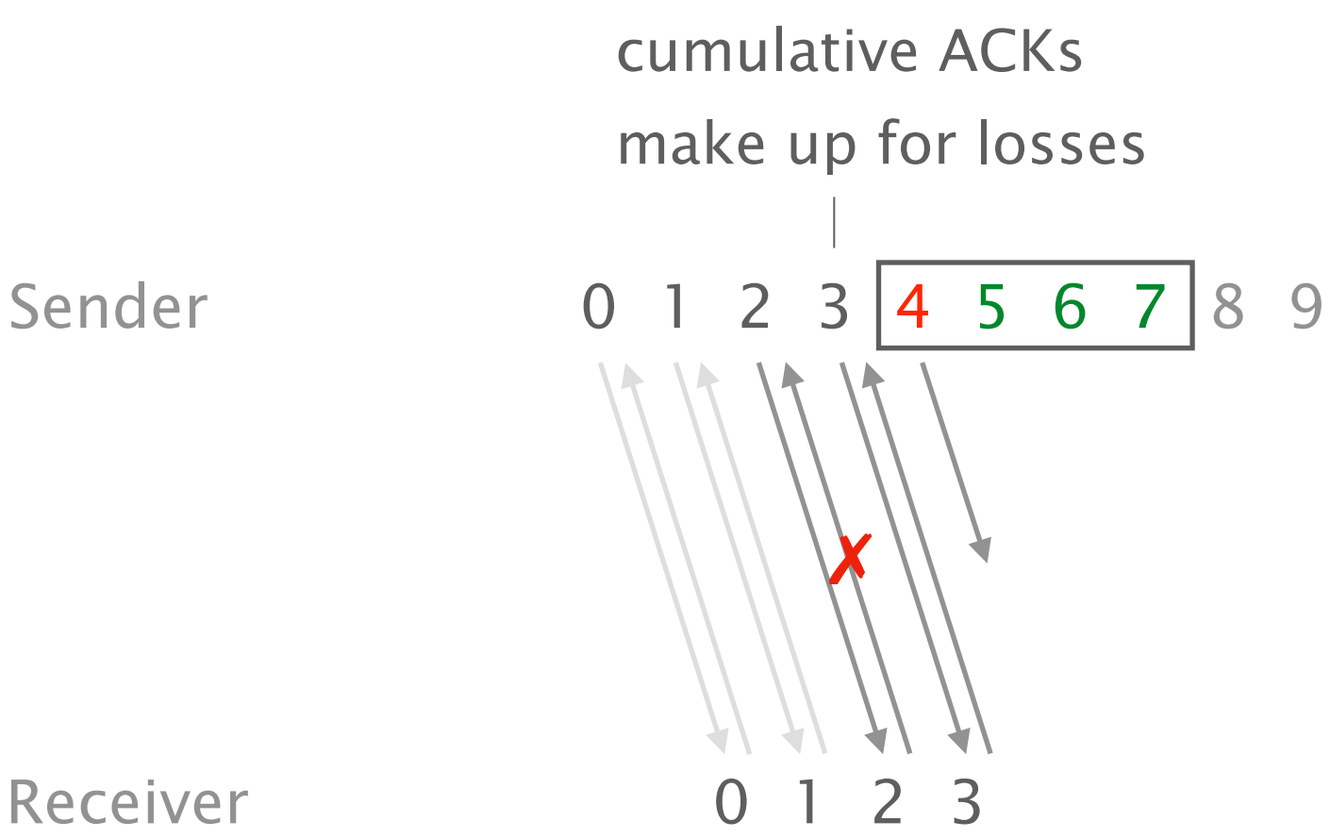
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

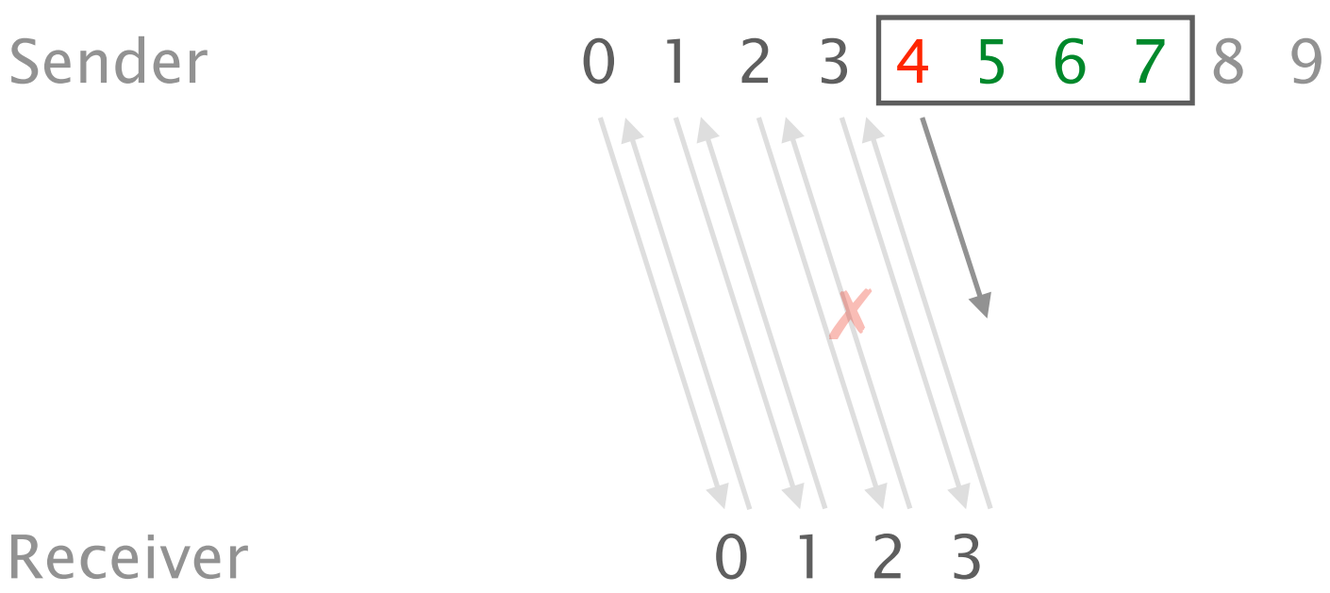
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

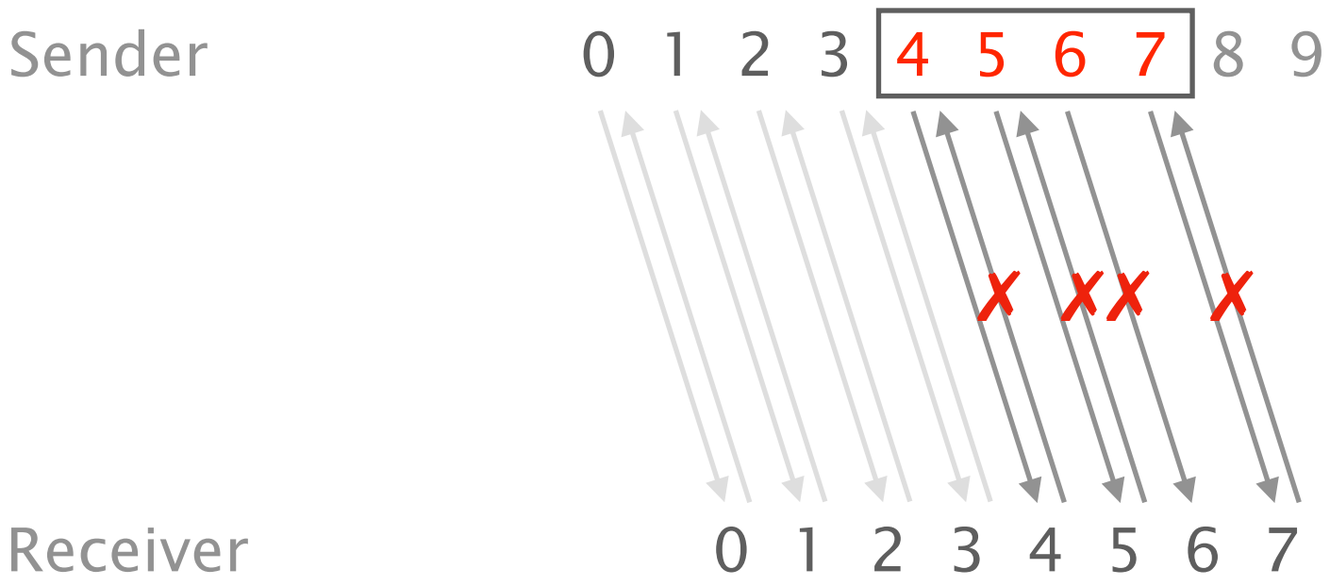
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

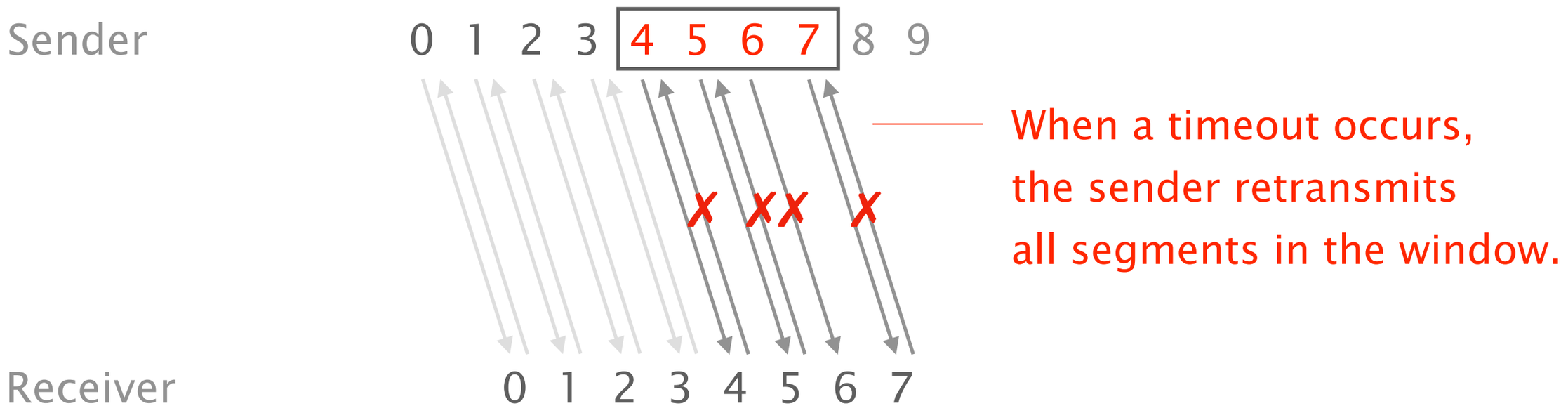
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

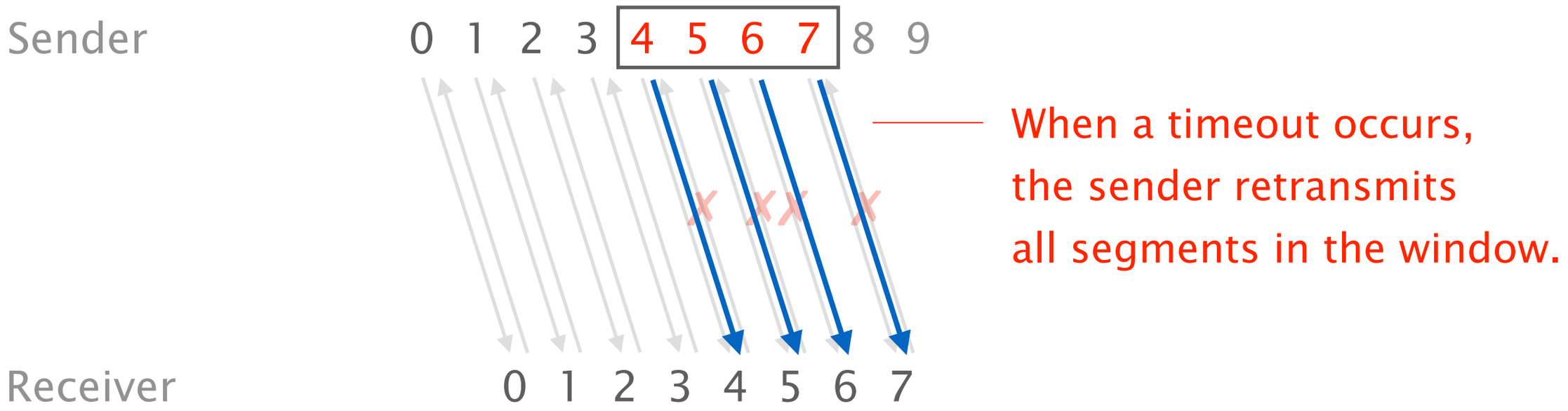
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

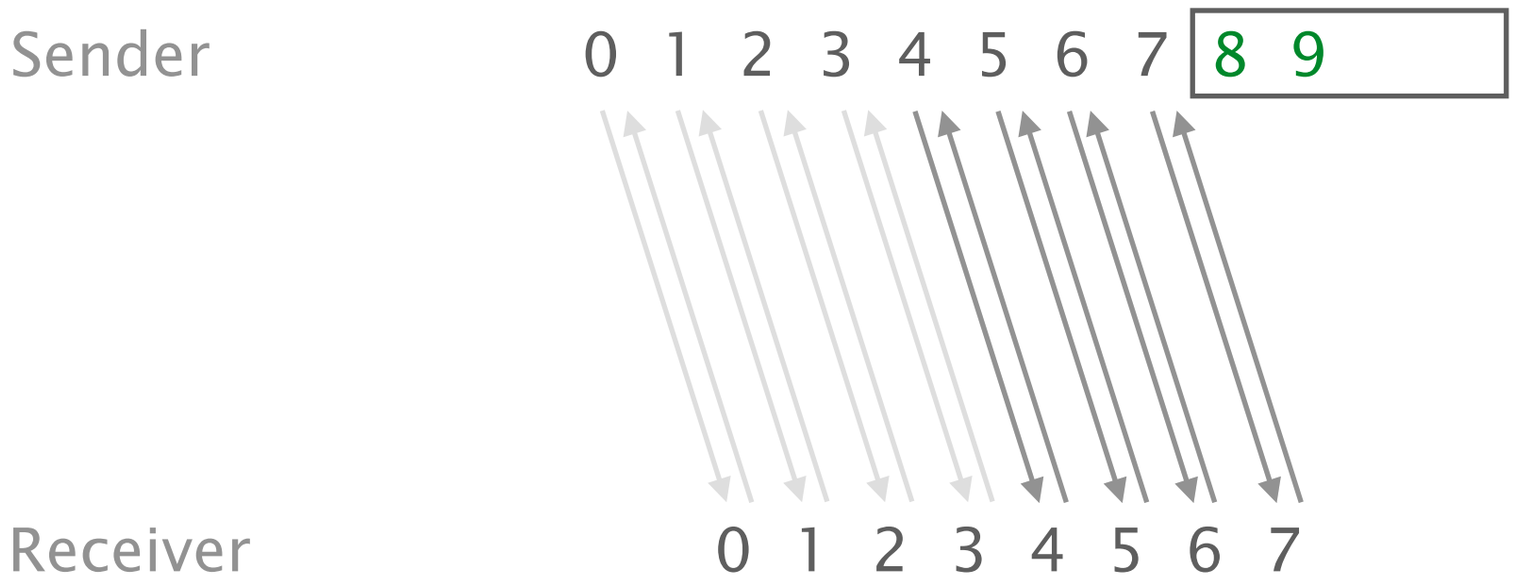
a sliding window, cumulative ACKs, timeouts and retransmissions



The Go-Back-N Protocol

a simple reliable transport protocol with

a sliding window, cumulative ACKs, timeouts and retransmissions



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

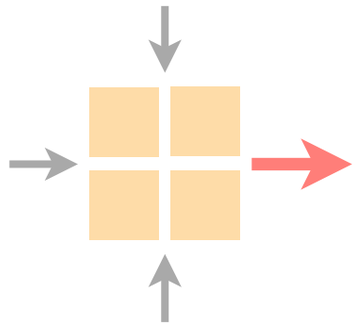
Don’t worry, we provide you with a code skeleton

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

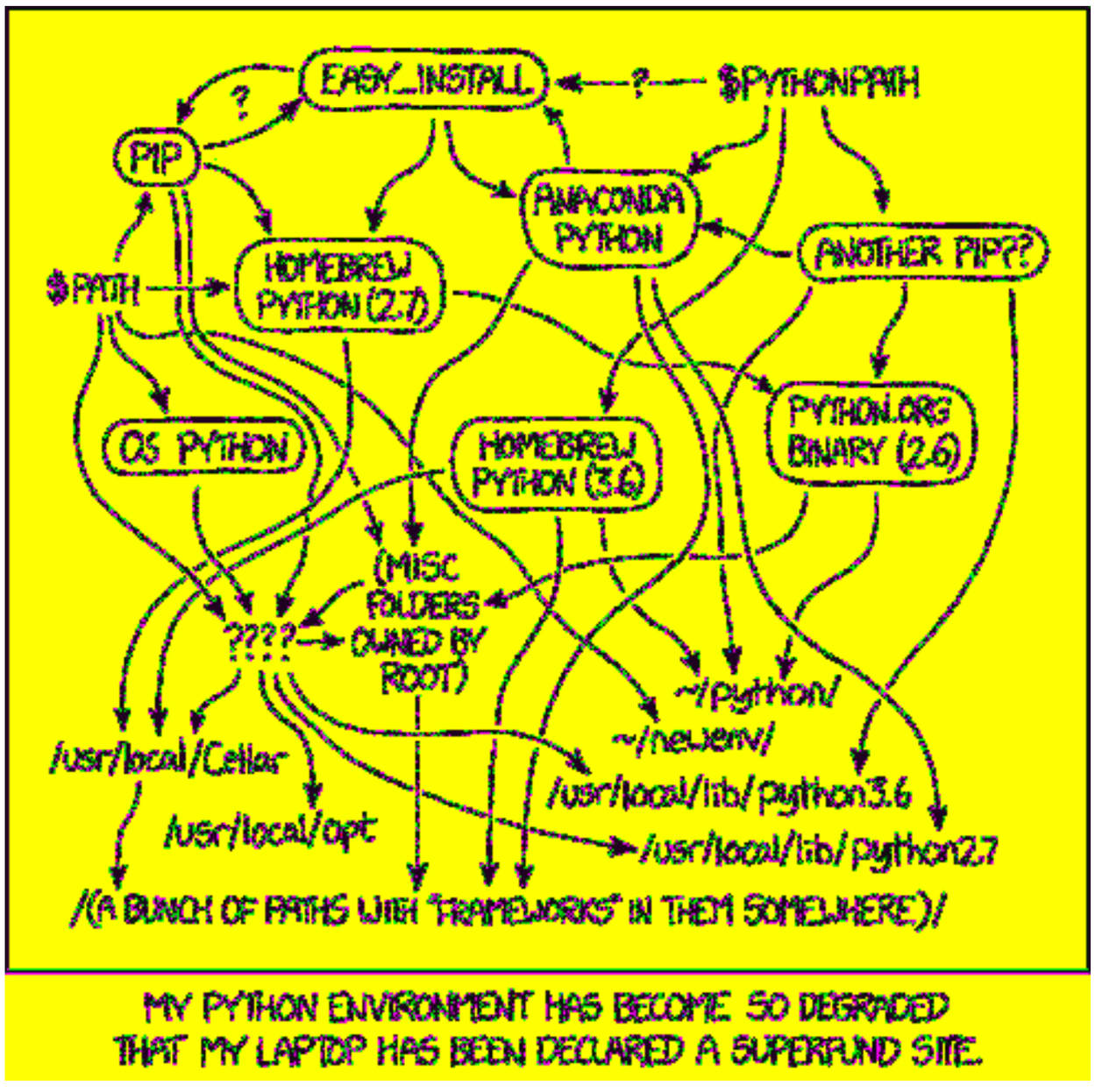
And a bit of Git

ETH Zürich - Communication Networks - Noah Hütter

Contents

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

Python



https://xkcd.com/353/

Python

Installation

Windows

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

Advanced/Refresh Guide
learnxinyminutes.com/docs/python3

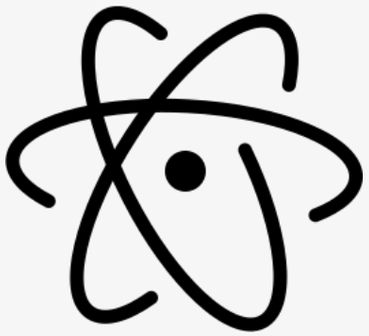
```
8 - 1   # => 7
10 * 2  # => 20
35 / 5  # => 7.0

# 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

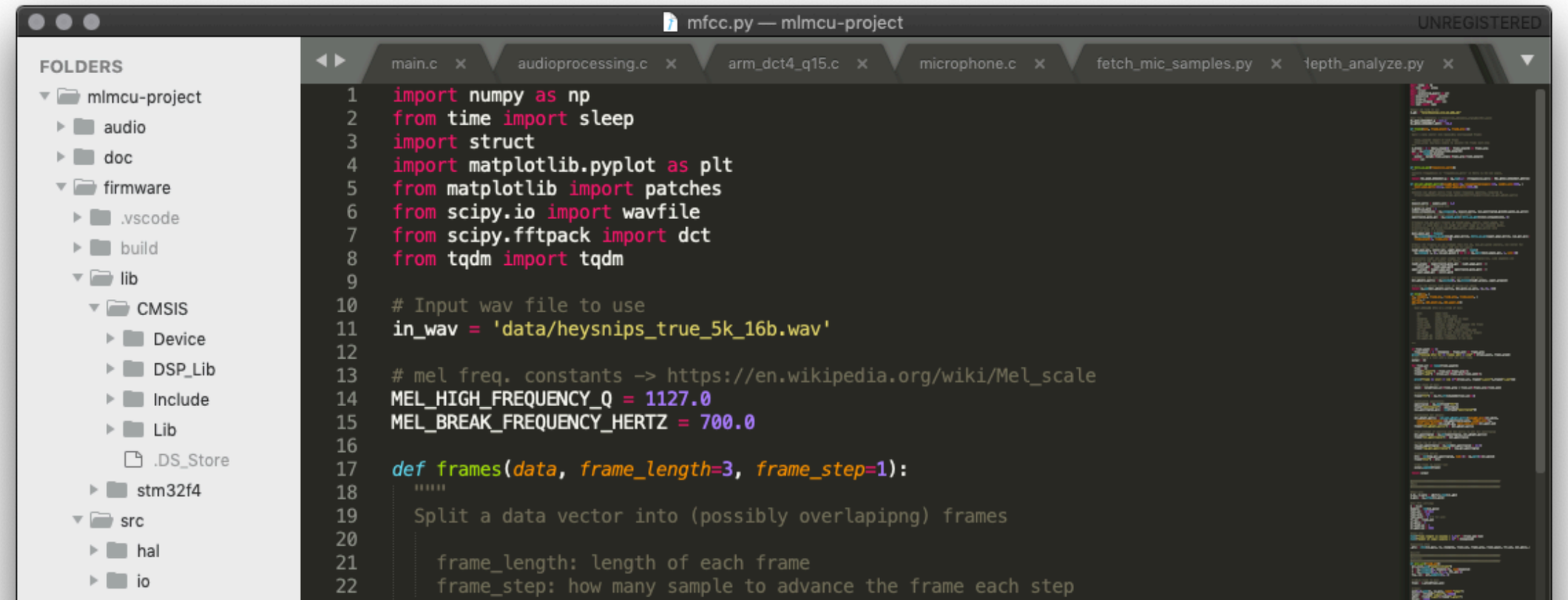
Visual Studio Code

Python

From Text Editors to IDEs

Sublime Text

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

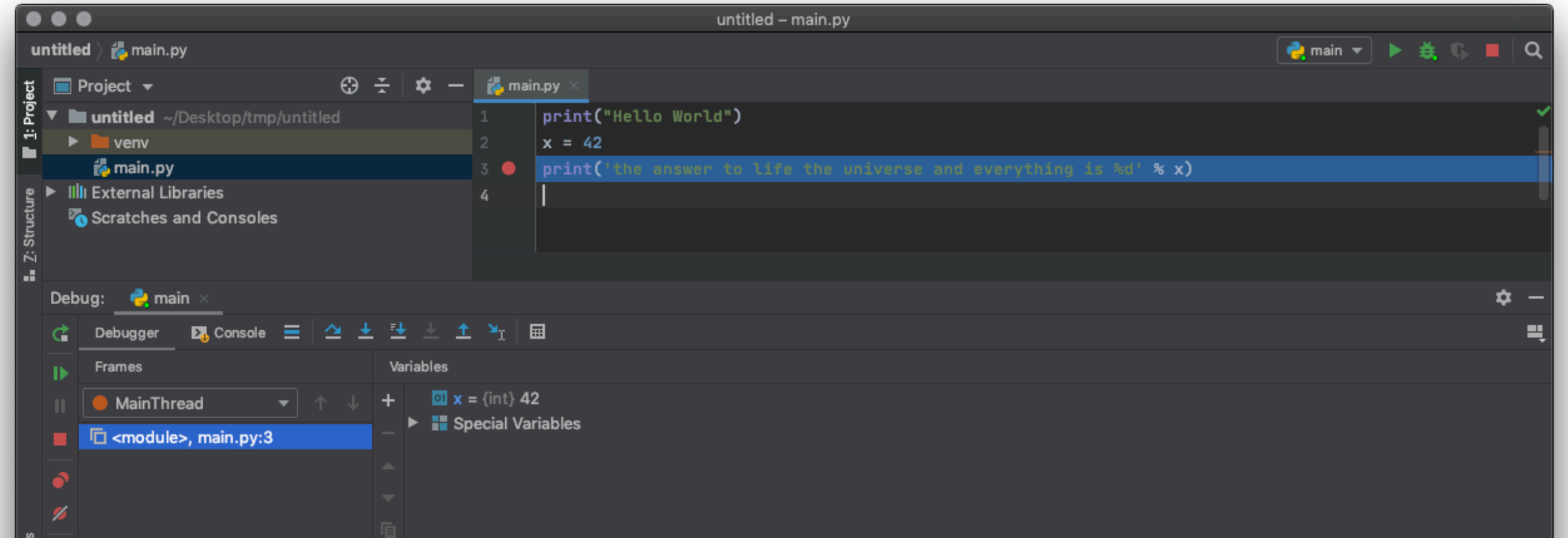


Python

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)

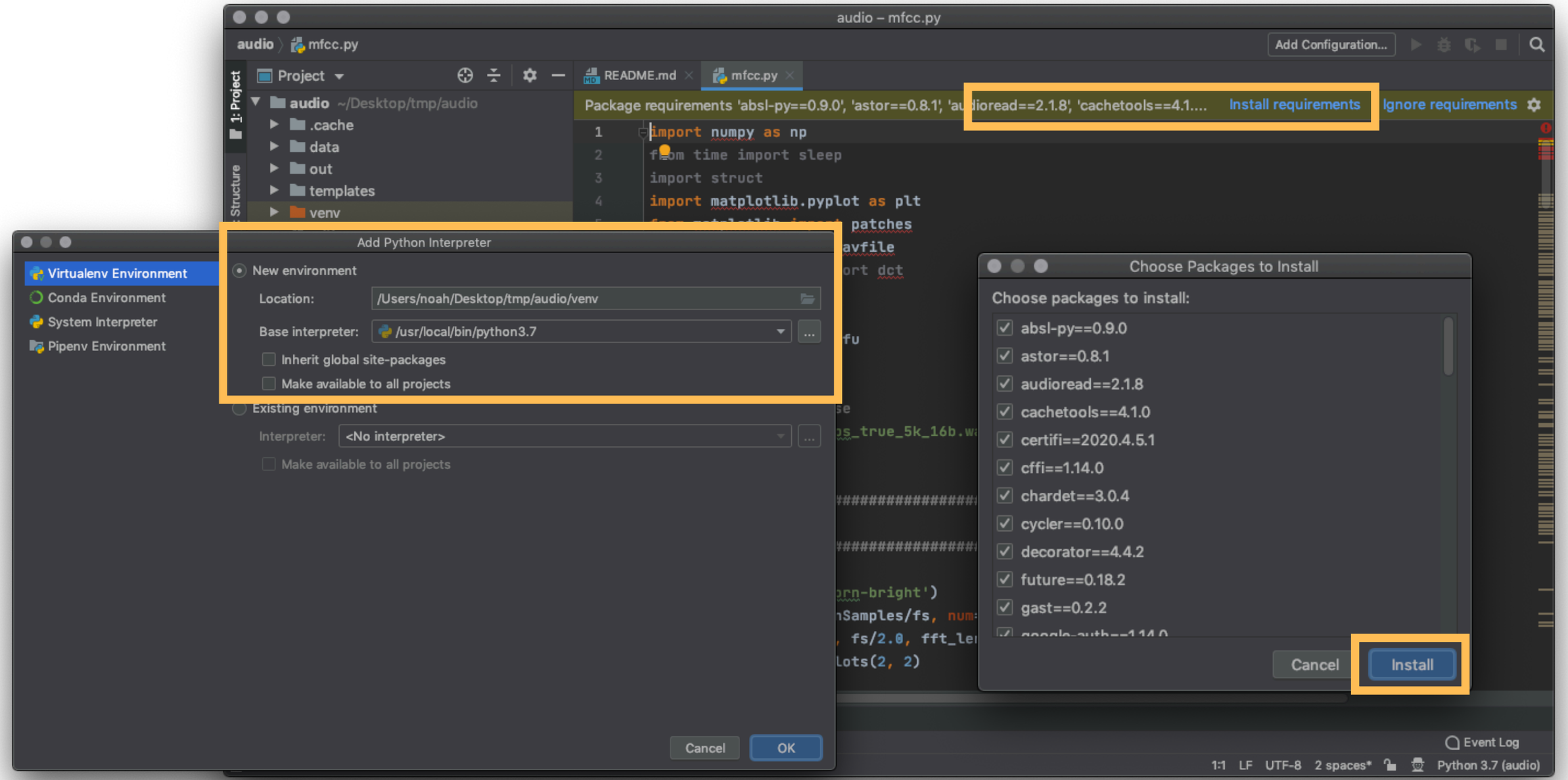
Helps to use correct Python version and packages.
https://realpython.com/python-virtual-environments-a-primer/

```
# 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

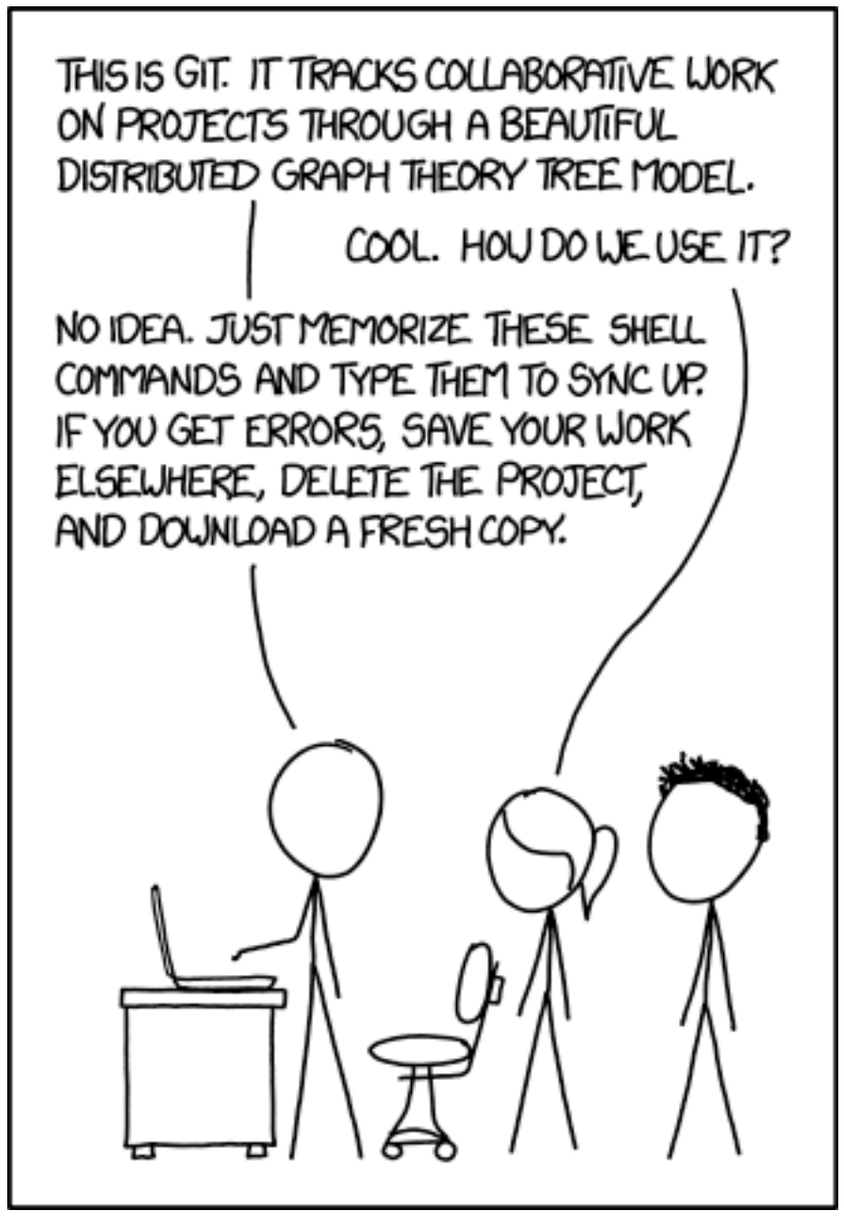
Virtual Environments (Advanced topic)

Supported by PyCharm



Git

Version Control



https://xkcd.com/1597/

Git

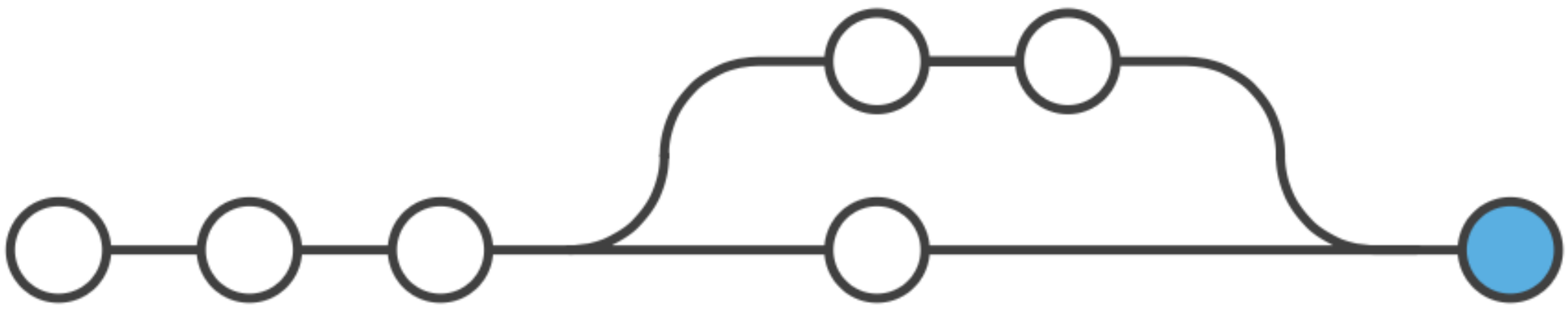
Tracks Changes in your Code

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)



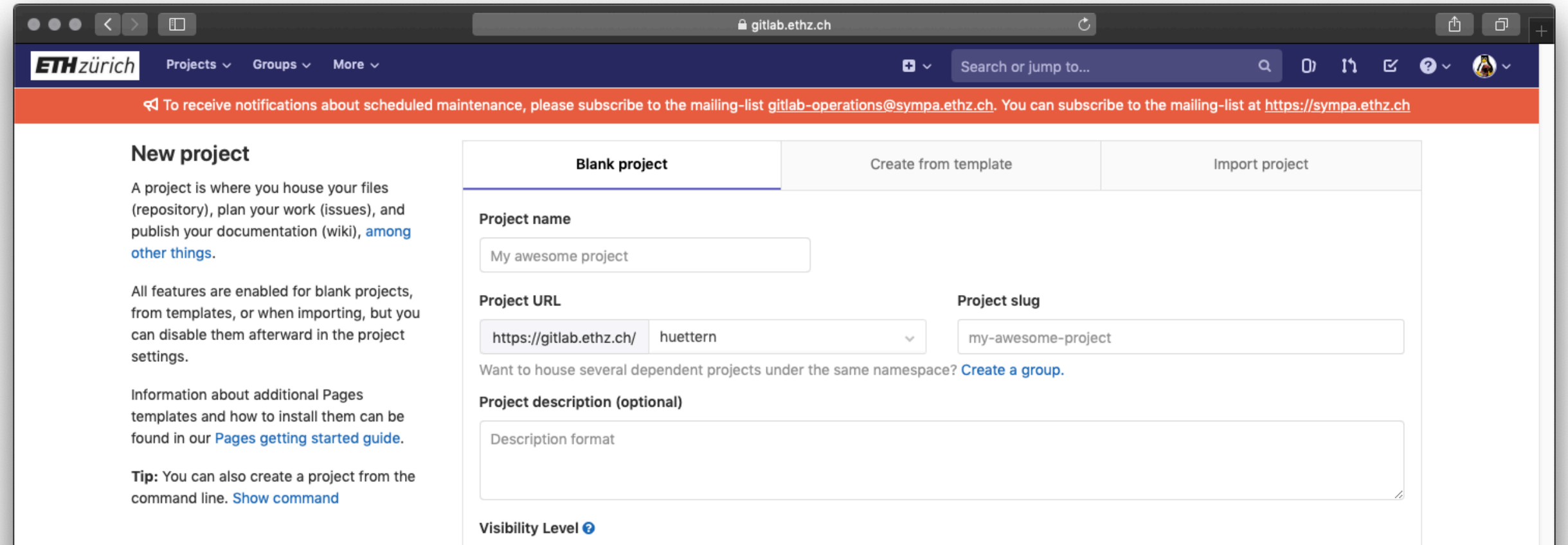
Git

Setup

1. Create a repository for your group

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

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



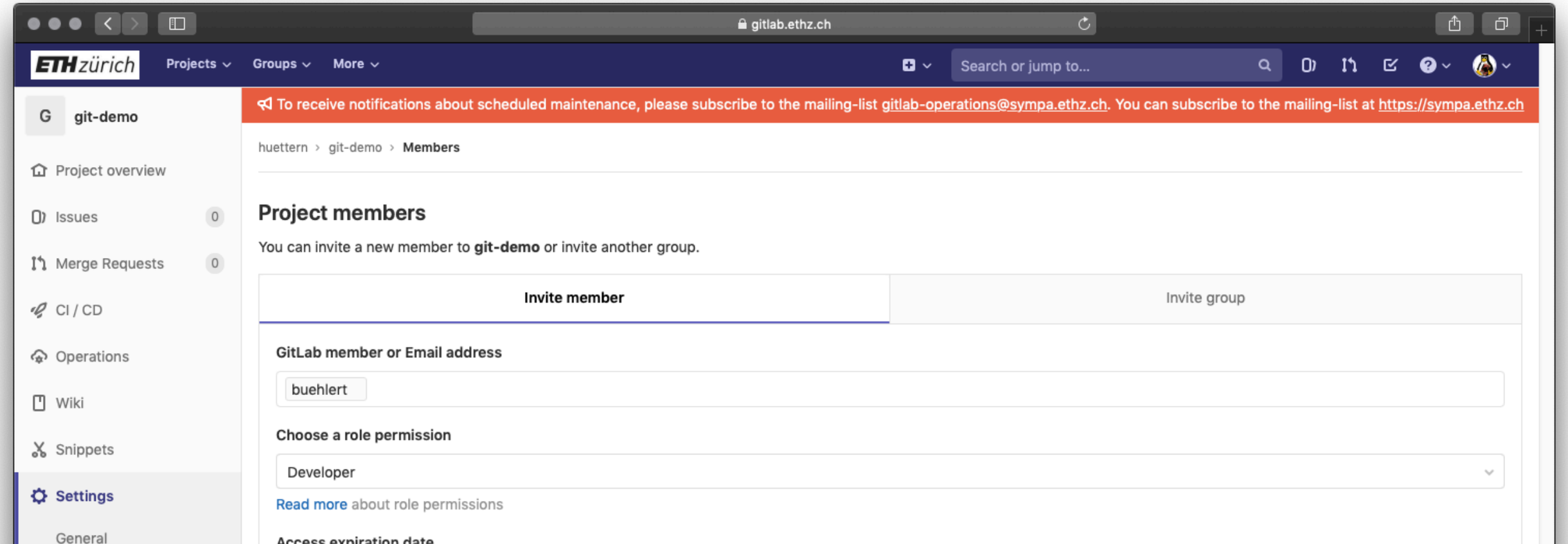
Git

Setup

2. Invite group members

Settings → Members

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



Git

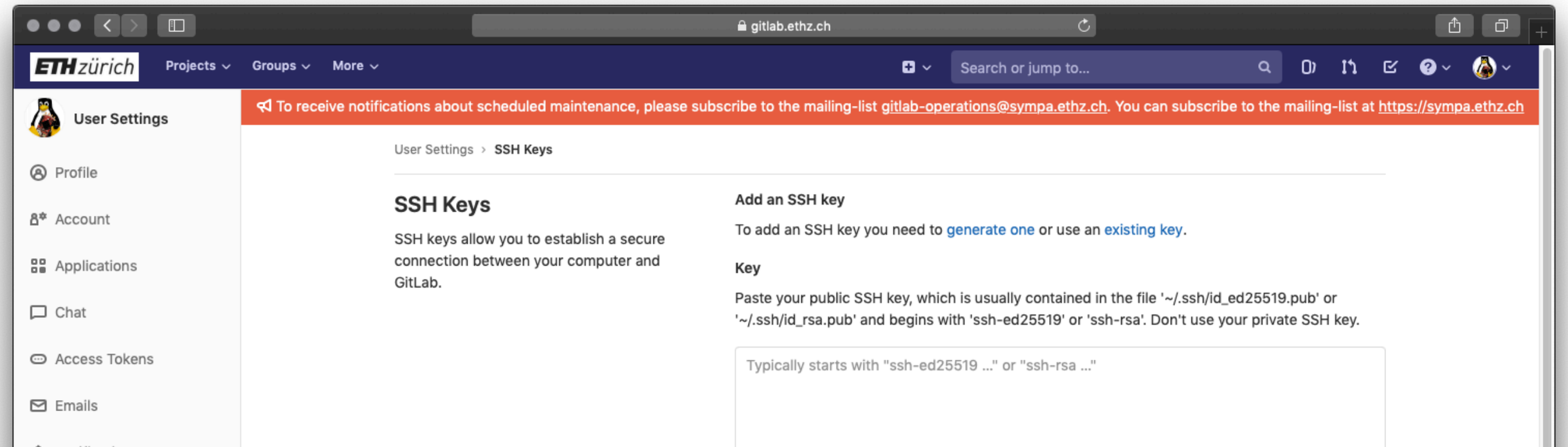
Setup

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.

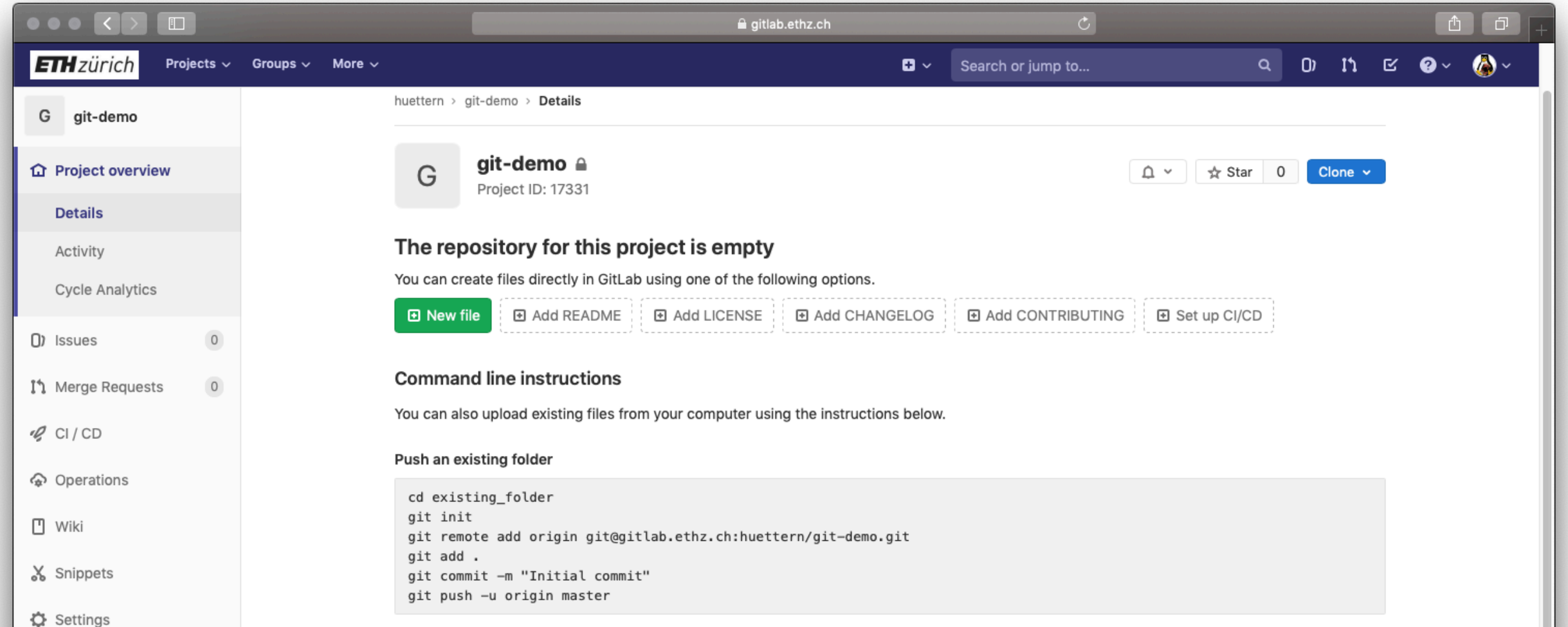


Git

Setup

4. Upload the project files to Gitlab

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



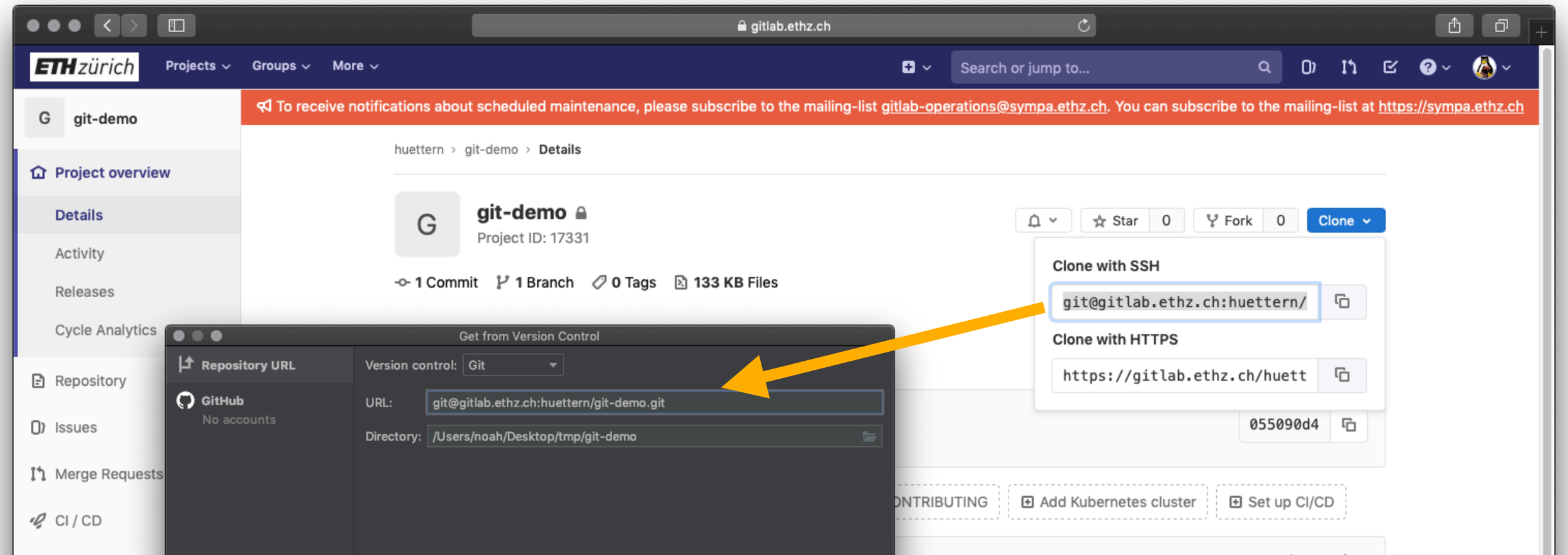
Git

Setup

5. Download the repository to your local machine

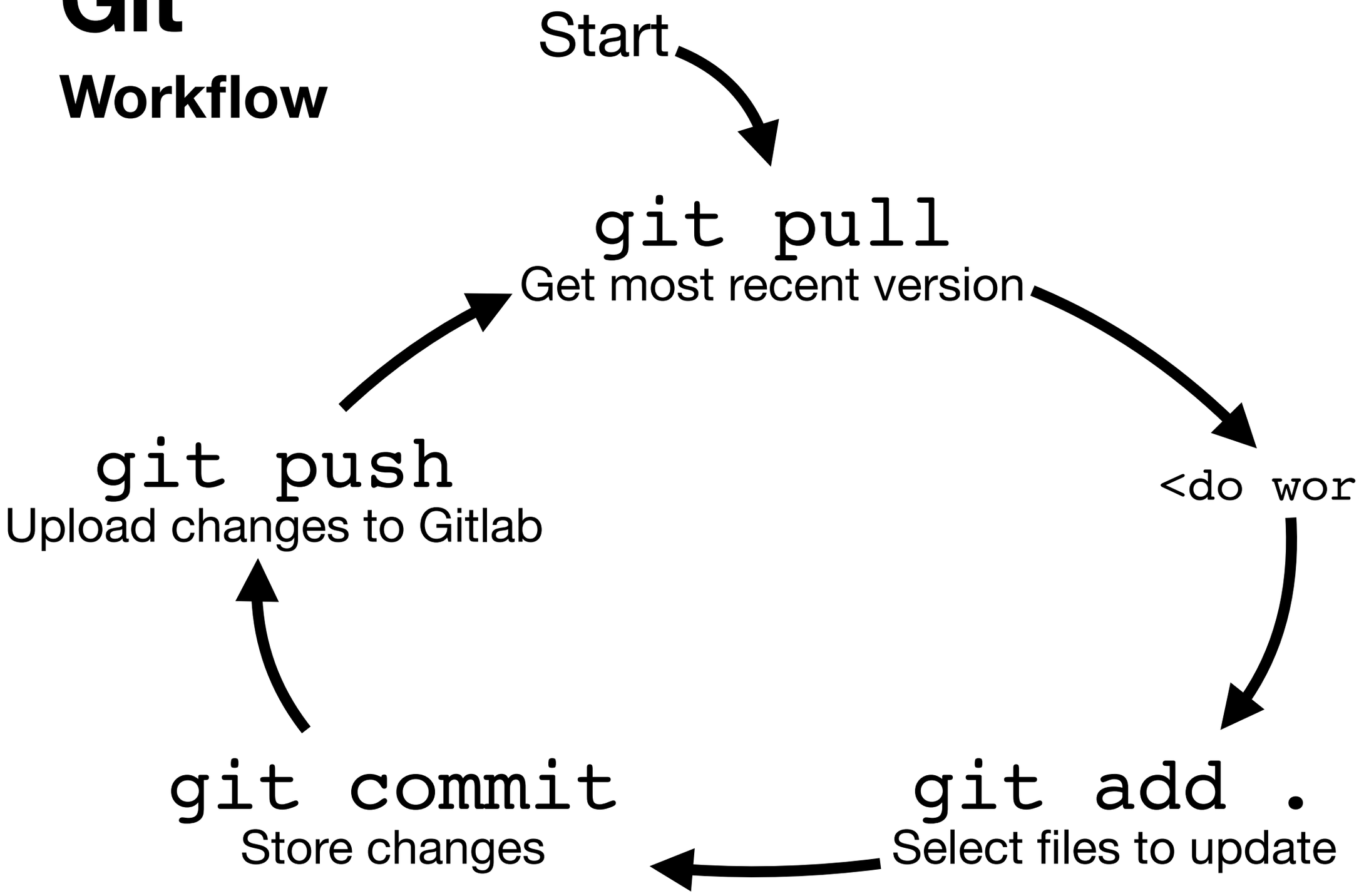
This way you can work on your machine without VM connection

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



Git

Workflow



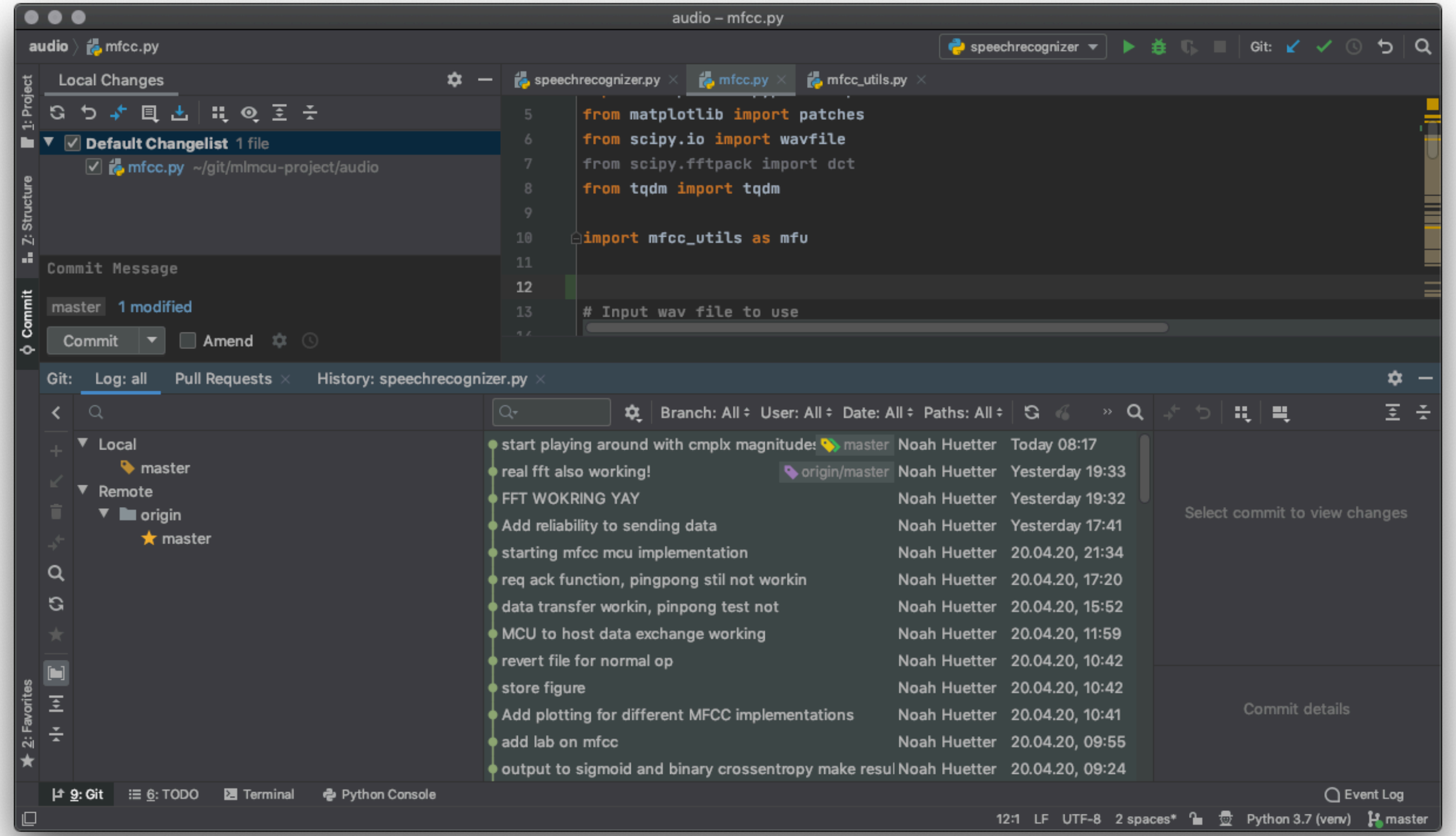
Git

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



Git

Tips and Tricks

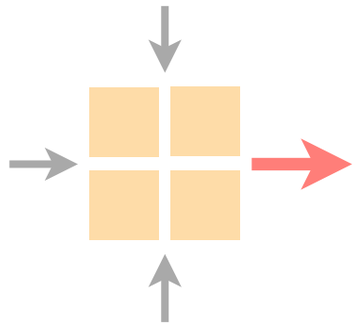
- No branching required for the assignment
- Run the git commands from the correct directory
- Always `pull` before you `push`

Cheat Sheet & Installation Guide

rogerdudler.github.io/git-guide

Communication Networks

Exercise 10



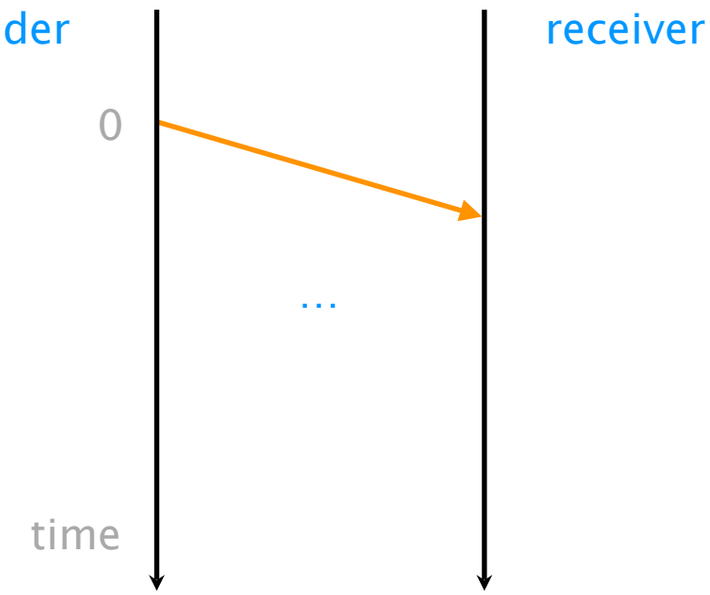
Wrap-up of the routing project

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)

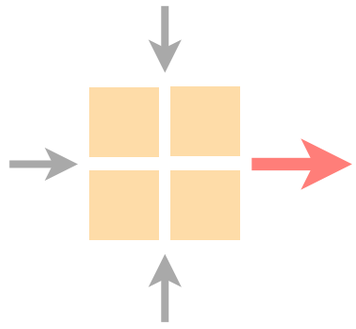
on a 10Mbps link with a

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