Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

# Advanced Computer Networking

**IN2097, Winter Semester 2024/25**

**Project - Designing a Software Router (10 credits)**

*Last updated: November 17, 2024 at 2:30pm*

The goal of this project is the development of a software router. We use the high-speed packet processing framework DPDK to implement a high-performance software router. To simplify the implementation of the router, we provide access to our testbed, where the software router can be developed and tested. The router will only implement IPv4 and associated protocols to simplify the router for the purpose of this exercise.

## 1 Academic Misconduct

We check your submissions for plagiarism. Participants violating the academic code of conduct will be excluded from the bonus system.

It is allowed and encouraged to discuss the assignment with other students. However, the programming itself has to be done by each student individually. Group work for writing the code is not allowed. Google, StackOverflow, and other Internet sources are allowed (as long as no license is violated). If your submission contains copied code, it has to be clearly marked as such, and the original source has to be referenced. For example, StackOverflow provides a share link. Use that to obtain a link, which then has to be added to your source code. In any case, you have to understand the code you submitted, which means you must be able to explain how it works.

See also code of conduct by our department:

- en: http://go.tum.de/103707
- de: http://go.tum.de/750259

## 2 Submission via git

For this project, we use git to handle the submissions. We use the same repository for our tutorials. If you do not have access to your repository yet, please read the tutorial instructions and perform the described steps to get access.

## 3 Testbed

Scientific testbeds are used to execute experiments, such as benchmarking hardware or software components. We aim to create reproducible experiments. Therefore, we automate the entire experiment workflow to minimize the chance of human error or misconfigurations during experiments. Our research group created a framework to manage testbeds called the *plain orchestrating system (pos)* [2]. pos ensures the creation of reproducible experiments using a specific experiment workflow. For the purpose of this lecture, we created a testbed providing a small network of four connected nodes (1 router, 3 clients) for every student. During the course of this project, you will get to know the pos framework, its experiment workflow, and its features.

### 3.1 Reservation of testbed resources

A reservation is needed to use the resources of the testbed. Therefore, we created a calender to enter the timeslots and the type of resources for a planned testbed usage. The calendar can be accessed from the CLI

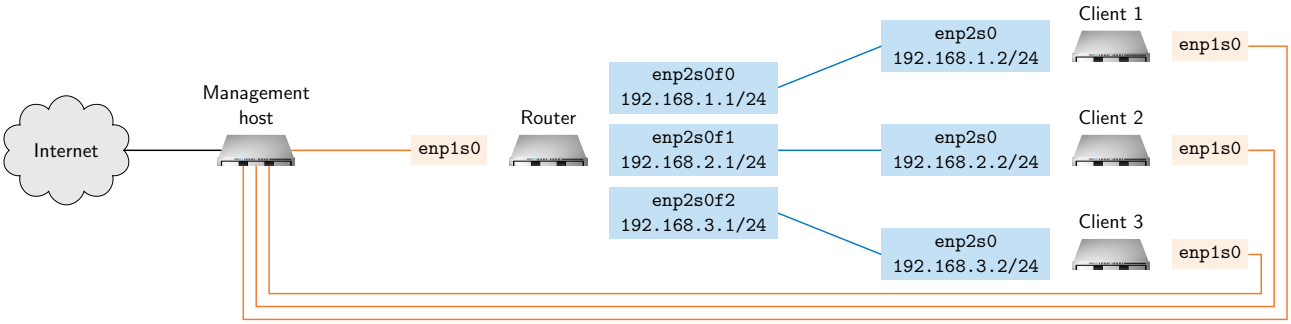


**Figure 1:** Testbed with management host, clients and router

(cf. Section 3.2) or via the web interface[1]. Multiple identical setups are available for testing, each setup consists of a topology with a single router and three clients as depicted in Figure 1. The router and clients with the same prefix (e.g., `i1`) are directly connected. Each testbed user has access to multiple setups.

**Fair use policy:** Every user can only allocate two future timeslots with a maximum length of 6 h. Please only allocate the minimum amount of resources necessary for testing to also give other people a chance to use the testbed.

## 3.2 Accessing the Management Node of the Testbed

The testbed consists of a single management node and multiple experiment nodes. Figure 1 shows the management host and the four experiment nodes used for implementing this project. The management node acts as the gateway to the testbed. All experiment nodes are connected to the management node via a management network (orange connections). This management network is separate from the experiment network (blue connections). This separation is necessary to avoid any impact of management traffic on measurements using the experiment network. The management node of the testbed can be accessed via SSH using the following command in Listing 1.

```
ssh -p 10022 gitlabUserID@svm0020.net.in.tum.de
```

**Listing 1:** Connecting to the management node (replace `gitlabUserID` with your actual user ID before trying to log in)

We use your GitLab user ID as the username and the SSH keys you uploaded to the LRZ GitLab for authentication. You can find your GitLab user ID in the web interface (cf. Figure 2). Make sure that at least one of your personal SSH keys known to GitLab is available on your local machine before trying to log in.

After logging in to the testbed host, you can access the pos CLI, the central tool to use the testbed. The pos CLI offers extensive documentation; just type in the pos command. An example can be found in Listing 2.

[1] https://svm0020.net.cit.tum.de/calendar

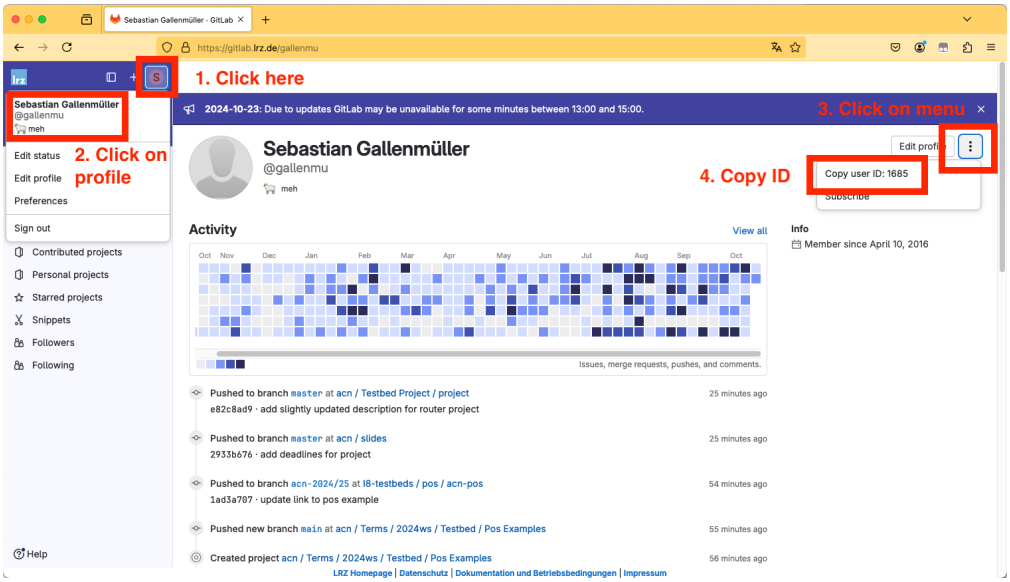


**Figure 2:** Location of Gitlab ID in the web interface

```
svm0020% pos
Usage: pos [OPTIONS] COMMAND [ARGS]...

  Plain-orchestrating service to access and manipulate the test nodes of this
  testbed.

  Quickstart:

  0. Look at available nodes:
      $ pos nodes list
      $ pos allocations list

  1. Allocate testnodes you want to use for one experiment:
      $ pos allocations allocate nodeA nodeB [...]

  2. Configure the nodes (per node):
      $ pos nodes image nodeA debian-stretch
      $ pos nodes reset nodeA

  3. Execute commands on the nodes (per node):
      $ pos commands launch nodeA -- echo $(hostname)

  You may use command prefixes, e.g. "pos n l" for "pos nodes list"

Options:
  --version              Show the version and exit.
  --color / --no-color
  -h, --help             Show this message and exit.

Commands:
  allocations  Define nodes taking part in an experiment.
  calendar     Testbed calendar
  commands     Execute commands on testbed nodes or roles.
  hooks        Configure hooks in posd that can be used as callbacks
  images       List available/add new images.
  jobs         Jobs (scripts) to be executed by pos at a given time
  nodes        Access testbed nodes or roles.
  roles        Group nodes into logical experiment roles.
```

**Listing 2:** pos CLI example on management node with help text output

## 3.3 Accessing the Experiment Nodes

Figure 1 shows the topology used for implementing this project. A router and three client nodes are available. The router is connected to each client via a separate connection. The router node has more RAM (8 GB) and three CPU cores compared to the client nodes, which have 4 GB of RAM and a single virtual CPU core. Every machine has a management interface called `enp1s0` to connect the respective experiment node to the management node. This interface provides SSH and Internet connectivity, i.e., this connection will not be interrupted when messing around with the other interfaces and DPDK.

Participants get their own set of experiment nodes for the project. The names of the experiments can be queried using the pos CLI, shown in Listing 3. This command outputs a table that lists the names (`id`s) of all available nodes and additional information, such as the configured image.

```
svm0020% pos nodes list

id          type  status       allocation                 image            updated

i0-client1  host  ERR booting  1685_231103_210856_811477  debian-bookworm  48h
i0-client2  host  booted       1685_231103_213724_568848  debian-bookworm  48h
i0-client3  host  unknown      None                       boot-local       6d
i0-router0  host  unknown      None                       boot-local       6d
```

**Listing 3:** pos command to list the available nodes of the testbed

To log in to a specific node, use the SSH command on the management node followed by the ID of the respective experiment node. Keep in mind that the node has to be booted before it can be accessed. Listing 4 shows several nodes. According to the printed output, only the experiment node with the ID `i0-client2` is currently in a booted state and can be accessed via SSH.

```
svm0020% ssh i0-client2
```

**Listing 4:** Logging in to a booted experiment node

To set up the nodes that are currently in a non-booted state, several steps have to be performed:

1. Allocate the respective node(s).
2. Configure an image to be booted.
3. Reset the machines to load the configured image.
4. Wait for the machine to become available.

Listing 5 lists the respective commands to perform the previously listed steps. We also provide a more extensive example script[2].

```
svm0020% pos allocations allocate i0-client1
Allocation ID: 1685_231105_214817_869983 (i0-client1)
Results in /srv/testbed/results/1685/default/2023-11-05_21-48-17_869983
svm0020% pos nodes image i0-client1 debian-bookworm
svm0020% pos nodes reset i0-client1
```

**Listing 5:** Allocating and preparing an experiment node with id `i0-client1`

[2] https://gitlab.lrz.de/acn/terms/2024ws/testbed/pos-examples.git

## 3.4 Rebooting Experiment Nodes

Nodes can be rebooted from the management node using the command in Listing 6. This also works if the connection between the experiment node and the management node is no longer available, e.g., if the NIC driver was removed.
**Important note:** All experiment nodes use ramdisks as storage; resetting a node erases everything that was written to this storage. Only your home folder on the management node is permanent.

```
svm0020% pos nodes reset i0-client1
```

**Listing 6:** pos command to reboot a node with id `i0-client1`

## 3.5 DPDK Framework

To implement your router, we provide a framework, containing DPDK and a simple forwarding application based on DPDK. We use a slightly adapted version of DPDK for this project. You must use this version for all of the problems of this project to get the bonus. You can get it using the command in Listing 7:

```
git clone https://git:glpat-MuwDUtiuYfQcno43swz_@gitlab.lrz.de/acn/terms/2024ws/testbed/dpdk-framework.git
```

**Listing 7:** Cloning the DPDK framework repository

## 3.6 Template Repository

The submission follows a specific file and folder structure. To simplify the creation of this folder structure, we provide a template that can simply be merged into your own repository. To merge the folder structure into your own personal repository, execute the commands of Listing 8.

```
git remote add template git@gitlab.lrz.de:acn/terms/2024ws/template.git
git remote update
git merge --allow-unrelated-histories template/router-project
```

**Listing 8:** Merging the router-project branch

After merging `commit` and `push` your changes.

# Problem 1 Setup — 1 credit

The deadline for this problem is **November 26, 2024, 4:00 PM**.

This exercise is designed to get to know the testbed and to set up the DPDK framework and the client machines, respectively. As part of this setup must be repeated in case of a node reboot, you create scripts to automate this step. Furthermore, a simple DPDK forwarding example is used and extended to test your setup.

Use the commands specified in Subsection 3.6 to create the folder structure required for submission.

**Default route** You are connected to an experiment node via SSH. Your SSH connection uses the default route installed on your local machine.

**a)** Why is it a horrible idea to remove this default route? Put your answer into a file named `answer.md` in the `router-project1` subfolder of your git repository.

**Experiment script** Create an experiment script that allocates all four experiment nodes, loads the provided parameter yaml files for the respective nodes, configures the image, and finally reboots all nodes with the configured image. After that, the experiment script should execute the `client.sh` and `router.sh` scripts in the `router-project1/node` folder for clients and router on the respective experiment nodes. These experiment node scripts are currently empty. The following subproblems will provide content for both scripts.

**Hint:** Have a look at the example script[2].

**b)** Put the final script (`experiment.sh`) in the `router-project1` subfolder of your git repository.

**Clients** The configuration for the client experiment nodes involves three steps:

1. Assign the correct IP addresses (see Figure 1)
2. Set the interfaces up
3. Configure the routes to their respective neighboring clients

These steps must be repeated after a reboot of a node. Create a script to automate these steps for each client individually. Use the Linux tool `ip` [1] for configuration (no credits for `ifconfig`). After executing the script, the node must still be reachable over SSH via the management interface.

The same steps must be executed on all three client nodes. However, the configured addresses differ slightly between clients. To handle this problem efficiently, pos offers the possibility to parameterize scripts on a per-node basis. Parameters can be defined in yaml files, configured on the management node (`pos allocations set_variables`), and queried on the experiment nodes (`pos_get_variable`). An example for such a configuration can be found in the example code at
`https://git:glpat-qGxMJZXcUPk1hvNWAjps@gitlab.lrz.de/acn/terms/2024ws/testbed/pos-examples.git`.

**c)** Create a script that configures the three clients (`client.sh`, `client1.yml`, `client2.yml`, and `client3.yml`) and put them into the `router-project1/client` subfolder in your git repository.

**Router** The steps to set up and compile the DPDK framework can be taken from the README file in the framework repository[3]. Keep in mind that a configuration of the interfaces via standard Linux tools is not possible after the interfaces are managed by the DPDK application. DPDK uses increasing numeric interface identifiers ordered by the PCI address of the device, i.e., the router uses the interface ids 0 to 2 if all VirtIO interfaces are bound to DPDK.

The DPDK setup must be repeated after a restart. Create a script to automate these steps so that a DPDK application can be started after running this script.

**Hint:** You can reboot any experiment node anytime from the management node to test your script.

[3] https://gitlab.lrz.de/acn/terms/2024ws/testbed/dpdk-framework.git

**Note:** Interfaces that are bound to the DPDK driver will **NOT** be listed in tools such as `ip` any longer.

**d)** Create a script that configures the router (`router.sh`) and put it into the `router-project1/router` subfolder in your git repository.

**Forwarder test** To test your setup, a simple unidirectional forwarding application (`fwd`) is provided. This application takes the incoming Ethernet frames from a source interface (e.g., `-s 0`), increments the MAC source address by one and sends it via a destination interface (e.g., `-d 1`).

To see if the forwarder actually works, use the tools `ping` and `tcpdump`. Run the forwarder on the router node and try to ping the client2 node from the client1 node on `eth1`, respectively. On the client2 node, run `tcpdump` on `eth1`.

**e)** Which kind of packets do you expect to arrive at client2 and what kind of packets do you actually observe. Briefly explain the observation. Put your answer into the previously created `answer.md`.

**Bidirectional forwarder** `fwd` only forwards unidirectionally, i.e., it forwards packets from a specified source interface port to a destination interface port. Extend the forwarder example to support bidirectional forwarding, i.e., forwarding from the source interface port to the destination interface port and vice versa.

**Hint:** By creating a multi-threaded forwarder this exercise can be solved with less than five lines of code.

**f)** Update the original `fwd.c` file and put it into the `router-project1` subfolder in your git repository.

Commit and **push** the contents of the `router-project1` subfolder in your git repository.

## Problem 2 Routing — 4 credits

The deadline for this problem is **December 19, 2024, 4:00 PM**.

The problem description will be released at a later point in time.

## Problem 3 Routing table — 3.5 credits

The deadline for this problem is **January 14, 2025, 4:00 PM**.

The problem description will be released at a later point in time.

## Problem 4 Measurement — 1.5 credits

The deadline for this problem is **January 30, 2025, 4:00 PM**.

The problem description will be released at a later point in time.

# References

[1] ip(8) - Linux man page. https://linux.die.net/man/8/ip.

[2] Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe, and Georg Carle. The pos framework: a methodology and toolchain for reproducible network experiments. In *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021*, pages 259–266. ACM, 2021.