NTNU - Department of Information Security and Communication Technology

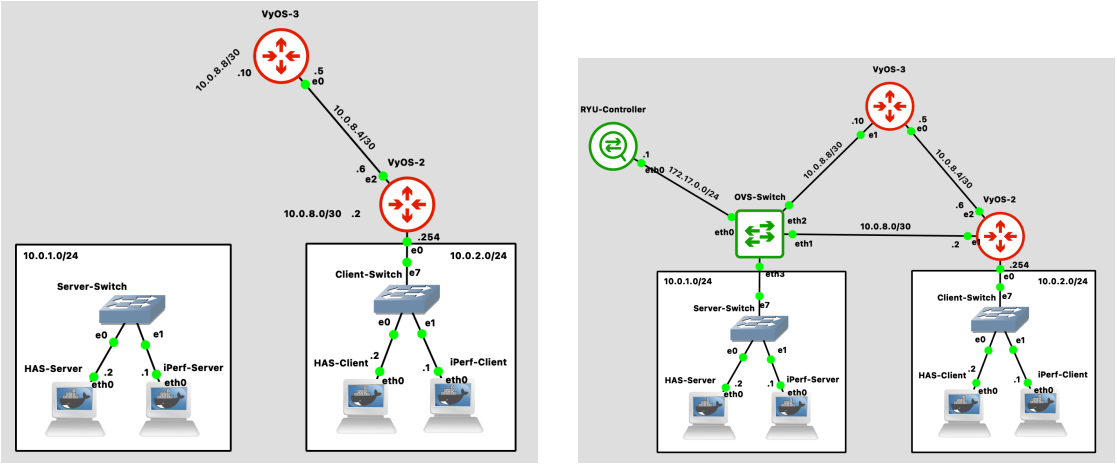
TTM4250 - Semester Assignment 1 (2025): Softwarized Networks

Introduction:

In this lab, you will get hands-on experience regarding network softwarization and SDN. In particular, you will replace a routera with a software switch (Open vSwitch, ovs). Additionally, you will add an SDN controller in order to make the network *programmable* in the sense that it will be able to react to network conditions in a dynamic manner.

After helping Anna to set up a network and its management infrastructure, Anna contacts you again regarding a new issue with the network: unfortunately, the router VyOS-1 has failed due to a hardware issue. Since she learned about network softwarization during her studies, Anna suggested using the failure of VyOS-1 as an opportunity to transition towards an SDN-based setup. Such a setup will not only lead to lower costs that stem from using standard servers, but will also allow a more dynamic re-routing of the different traffic types based on their respective bandwidth consumption.

The network state after the failure of VyOS-1 is displayed in Figure 1a. The goal of the SDN upgrade consists of replacing VyOS-1 with an instance of the Open vSwitch (ovs) software switch and the Ryu SDN controller as shown in Figure 1b. In this exercise, you will first help Anna to restore the original functionality of VyOS-1 to make the network operational again. Interestingly, replacing the router with a switch is functional, but only due to small modifications to the previous configuration. Initially, we ant to identify what features make the switch application function. We will then adjust the configuration and turn the ovsswitch and the Ryu controller into a router with gateway functionality. Finally, you will implement a controller module that will monitor the bandwidth consumption of the applications in the network and re-route video traffic if the iperf traffic exceeds a threshold.



(a) Network topology after the failure of VyOS-1.

(b) Target network topology after SDN upgrade. OVS switch interface names might be different

Figure 1: Network topology, before and after the SDN upgrade.

For this lab assignment, you will need to connect to a virtual machine (VM) available in an OpenStack cloud. Each group has one active VM. You need to have your own local installation of gns3 be able connect to the server using ssh, you can find a guide on how to do this here `https://ttm4250.iik.ntnu.no/latest/wiki/Setup/`. You will be working in the GNS3 virtual environment.

For the lab, we will be primarily be using OpenFlow 1.3.

Below, you can find helpful material that might help with the exercises.

- NTNU TTM4250 Wiki
  - `https://ttm4250.iik.ntnu.no/latest/wiki/`
- Open vSwitch (ovs)
  - General introduction: `https://docs.openvswitch.org/en/latest/intro/`
  - Starting ovs: `https://docs.openvswitch.org/en/latest/intro/install/general/#starting`
  - Man page for `ovs-ofctl`, the CLI tool for interacting with ovs: `https://man7.org/linux/man-pages/man8/ovs-ofctl.8.html`
  - Man page for `ovs-vsctl`, the CLI tool for interacting with the ovs daemon: `http://manpages.ubuntu.com/manpages/xenial/man8/ovs-vsctl.8.html`
- Ryu SDN Controller
  - Getting started: `https://ryu.readthedocs.io/en/latest/getting_started.html`
  - Writing custom modules / apps: `https://ryu.readthedocs.io/en/latest/writing_ryu_app.html`
  - Exemplary implementations of controller modules: `https://github.com/faucetsdn/ryu/tree/master/ryu/app`
  - Exemplary controller module for traffic monitoring: `https://osrg.github.io/ryu-book/en/html/traffic_monitor.html`
  - Structure and options of `OFPMatch`, used to compose matching rules: `https://github.com/faucetsdn/ryu/blob/master/ryu/ofproto/ofproto_v1_3_parser.py#L727`
  - Structure and options of `OFPActionSetField`, used to compose actions: `https://github.com/faucetsdn/ryu/blob/a394673993468ad3ee6cd784eb0d52aab967a8b3/ryu/ofproto/ofproto_v1_3_parser.py#L3356`
  - Documentation and examples regarding OpenFlow messages: `https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html`

Task 1.1 Integration of SDN components into the existing infrastructure

In this task, you will integrate the ovs and Ryu components into the network and restore its original functionality, i.e., providing connectivity between servers and clients according to a static routing scheme.

a) **Integration of ovs**

As a first step, you will familiarize yourself with ovs and restore the original functionality without an SDN controller.

In GNS3, you can add the new `ovs` component to the topology and use it as a replacement for VyOS-1, connecting it to the remaining components according to Figure 1b. For now, you can leave the `eth0` interface disconnected since it will be used for the controller connection later.

Open vSwitch is already installed in the `ovs` container. Consult the documentation at `https://docs.openvswitch.org/en/latest/intro/install/general/#starting` to find out how to start the `ovsdb-server`, initialize it via `ovs-vsctl`, and start the `ovs-vswitchd` daemon. High-level descriptions on how to do this is available in the wiki `https://ttm4250.iik.ntnu.no/latest/wiki/OVS/`

Note: You will only need three commands for this step. The setup of the PATH environment variable and the other steps leading to the `ovsdb-server` section are already taken care of.

Note: For this exercise, we do not use ovs's SSL support. Hence, you can omit the parameters `--private-key`, `--certificate`, and `--bootstrap-ca-cert` when launching `ovsdb-server`.

Provide the corresponding commands in your report.

Set up a bridge `br0` according to `https://docs.openvswitch.org/en/latest/intro/install/general/#validating` and add the three ports `eth1-eth3` to it. These are the ports that will be configured and programmed later. Provide the corresponding commands in your report.

Since we have replaced a router (L3) with a switch (L2), the default gateway (10.0.1.254) of the servers (iperf_sever, has_server) that were connected to R1 is gone. Nevertheless, we need to make sure that all traffic originating from the two servers is sent out via their primary interface (eth0). This can be achieved by adding a default route as follows:

- Right-click the machine in GNS3, select "Edit Config".
- Add "`up ip route add default dev eth0`" at the bottom of the interface configuration.
- Comment out the gateway definition using "", so that there is no conflict between the default routes.
- Save.

Please check if this is configured, and if not make sure to have this included.

Similarly, VyOS-2 will need an interface-based route to the server-subnet 10.0.1.0/24. Add a static route on VyOS-2 using the command in configure mode:

```
set protocol static route 10.0.1.0/24 interface eth1
```

With the ovs ports set up, you need to configure the desired routing behavior by installing appropriate rules in the data path. This is achieved using the `ovs-ofctl` tool, which is introduced in the wiki under OVS and OpenFlow. ovs-ofctl is further documented at `https://man7.org/linux/man-pages/man8/ovs-ofctl.8.html`.

First, clear the flow-table of `br0`. Then, add appropriate rules to achieve the following behavior, using one rule per bullet point:

- All ARP packets shall be flooded, i.e., they shall be output from all ports except the one via which they were received.
- IP packets destined for the `10.0.1.0/24` subnet shall be output via the interface of the ovs node connected to Server-Switch.
- IP packets destined for `10.0.2.1` shall be output toward VyOS-2. Additionally, the destination MAC address of the packets shall be rewritten to the MAC address of VyOS-2's `eth1` interface. This can be achieved via the `mod_dl_dst` action.

– IP packets destined for `10.0.2.2` shall be output toward VyOS-3. Additionally, the destination MAC address of the packets shall be rewritten to the MAC address of VyOS-3's `eth1` interface.

Note: For introductions to building flows, see the wiki. For additional information on possible match fields and actions, you can consult the corresponding man pages at `https://man7.org/linux/man-pages/man7/ovs-fields.7.html` and `https://man7.org/linux/man-pages/man7/ovs-actions.7.html`, respectively.

Note: To verify that the desired rules are actually installed in the data path, you can use the `show` and `dump-flows` commands of `ovs-ofctl`.

Provide the `ovs-ofctl` commands in your report.

b) **Verification of Forwarding Behaviour** Explain the general functionality of ARP and discuss the necessity of the MAC address rewrites added in the flow-rules. What happens if you just output the packets towards the `10.0.2.0/24` range without rewriting the MAC addresses?

If you attempt to restart the HAS- and iPerf-clients while capturing the traffic between OVS and VyOS-2/3, you should see application data as well as initial ARP-discovery.

Explain how is a packet forwarded from end to end (e.g. from has_server to has_client) using traces and node configurations. Specifically, describe how L2 and L3 addresses are used through the network and present any unexpected behaviour.

Note: You may want to look at how ARP requests are forwarded and responded to. Some of the VyOS-2 and 3 interfaces may include hints.

Note: Remember to turn off packet captures on links where you are done sniffing, to not bloat the disks of the VMs.

c) **Connecting the Ryu SDN controller**

Having completed the above steps, the original behavior of the network has been restored. However, to leverage the full functionality of SDN, you will need an SDN controller that can dynamically modifies the flow table of ovs. The goal of this task is to add the Ryu SDN controller to the network, connect it to ovs, and put it in charge of adding flow rules without changing the current behavior on the data plane.

First, ensure a `ryu` node to the GNS3 topology, configure appropriate IPs for the `eth0` interfaces of Ryu and ovs (e.g., 172.17.0.1 and 172.17.0.2, respectively), and interconnect them.

The IPs of OVS and Ryu can be set by right-clicking the component, selecting edit-config and uncommenting/setting the IP-address statically for the interfaces in question. Do not configure IP addresses for the remaining interfaces of OVS.

An installation of the controller is located in the `ryu/bin` folder of the Ryu container. You can start a controller instance that does nothing except the initial connection establishment and connection keepalive as follows.

```
cd ryu/bin
ryu-manager --verbose
```

Note: The `--verbose` flag can come in handy when debugging your custom controller module later.

Using `ovs-vsctl` (cf. `http://manpages.ubuntu.com/manpages/xenial/man8/ovs-vsctl.8.html`) inside the ovs container, configure ovs to use the newly added controller (the controller uses port 6653) and confirm that it is connected.

Your report should include the commands for setting up the controller connection and showing the status of the controller connection. Additionally, use wireshark to observe the link between ovs and Ryu while you set up the controller connection. What kind of messages of the OpenFlow protocol can you see? List at least three types of messages. What is their purpose? You can consult `https://web.archive.org/web/20200220214743/http://flowgrammable.org/sdn/openflow/message-layer/#tab_ofp_1_3` for detailed protocol information.

d) **Integration of the Ryu SDN controller**

At the moment, flow table entries have been set in the ovs by executing `ovs-ofctl` commands. However, the controller should be in charge of modifying flow rules. Hence, the next step is to create a custom controller module that will clear the flow table and populate it with the rules from the previous task.

Check how to integrate custom controller applications into Ryu and how to launch Ryu with such a module at `https://ryu.readthedocs.io/en/latest/writing_ryu_app.html`. Use the `--verbose` flag of `ryu-manager` to get detailed output, e.g., information on what module will handle what kind of messages.

Browse through the example applications at `https://github.com/faucetsdn/ryu/tree/master/ryu/app` to familiarize yourself with the functionality of Ryu and the structure of its applications. For instance the `add_flow` function implemented in `https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_switch_13.py` could be utilized for the current goal of populating the flow table. Similarly, the mechanisms around `_state_change_handler` in `https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_monitor_13.py` might come in handy for performing tasks upon connecting to a new switch.

Based on your research, implement a Ryu application that does the following: when connecting to a new switch, clear its flow table. Then, install the four flow rules listed in the previous task.

Note: In order to verify that your application works as intended, you can compare the output of the `dump-flows` command of `ovs-ofctl` when you run your application to its output when you manually added the flows in the previous task.

Provide the implementation of your controller application as a separate python file and describe how it achieves its goals. Furthermore, observe the exchange of OpenFlow messages during the removal / addition of flow table entries. What information is transported? What kind of OpenFlow message types are used?

e) **Ryu SDN controller as a router**

Since ovs (L2) has replaced a router (L3) we lost some functionalities that routers provide, for example a router replies to ARP requests for IP addresses in its network when requested from other networks. In order for ovs to become a full replacement of VyOS-1, we want those functionalities to be implemented in the Ryu controller. Therefore, please adjust router and server configurations to make sure that packets are forwarded to designated next hops and not via interfaces.

Servers:
Comment the "`up ip route add default dev eth0`" out of the network configurations and use the normal "gateway" definition.
VyOS-2:
`del protocol static route 10.0.1.0/24 interface eth1`

VyOS-2 and 3:
`del interface ethernet eth1 ip enable-proxy-arp`

Additionally, we need to delete all static flows that were added to ovs. Mention in your report how did you do that.

Now modify the controller application so that it functions as a router. The controller should maintain an ARP table where it stores MAC address to IP address mapping for the hosts in the network, as well as for the neighbouring networks. The controller should also work as the gateway for the hosts in the network `10.0.1.0` and hence respective flows should be added to ovs. Include the Python file in your report and also provide traces and/or controller output messages to show that the ARP requests are forwarded to the controller and the replies are also sent by it.

Task 1.2 Extending controller functionality

The route via VyOS-3 is expensive and it is not feasible to run the HAS traffic there permanently, especially since there is enough capacity on the VyOS-2 route when iPerf traffic is not active. Hence, Anna's idea is to use SDN to create a bandwidth-aware routing mechanism that reactively moves HAS traffic to the VyOS-3 route whenever it detects that the iperf portion of the traffic exceeds 1.5 Mbps and therefore impacts the video traffic too much.

In this part of the exercise, you shall work towards such a bandwidth-aware routing mechanism. First, the custom controller module will be extended to collect flow statistics. Then, these statistics will be used to infer per-flow and therefore per-application bandwidth usage. Finally, the bandwidth data will be used to dynamically re-route the HAS traffic.

a) **Add statistics collection**

Check the protocol information at `https://web.archive.org/web/20200220214743/http://flowgrammable.org/sdn/openflow/message-layer/#tab_ofp_1_3` for suitable OpenFlow messages regarding statistics collection. Based on your findings, figure out how to exchange the appropriate messages using Ryu. The main functionality to add to your module consists of sending statistics requests periodically (e.g., every 2 sec) and registering event handlers to process the replies.

Document the kinds of statistics that are available in the different statistics messages in your report. How could they be used for the goal of determining the current bandwidth usage of iperf?

Refer to `https://osrg.github.io/ryu-book/en/html/traffic_monitor.html` for an exemplary monitoring module that implements a superset of the desired functionality and integrate the relevant parts into your application. Functions of particular interest include `_monitor`, `_request_stats`, and `_flow_stats_reply_handler`.

Provide the implementation of your controller application as a separate python file and describe how it performs statistics collection.

b) **Use collected statistics to infer bandwidth**

With regular statistics collection in place, you would like to calculate and keep track of the per-flow bandwidth consumption on the switch. Think about appropriate data structures to store and map flow statistics over time.

Implement the per-flow bandwidth monitoring with a monitoring interval of 2 seconds and let your application log the current bandwidth of each flow to the console via the `self.logger.debug` function. It shall also output a warning whenever the bandwidth of the iperf flow (you can assume that this is the flow whose destination IP address is equal to `10.0.2.1`) exceeds a bandwidth of 1.5 Mbps.

Note: Keep in mind the units (byte vs bit) and potentially necessary conversion when using values from the flow statistics.

Provide the implementation of your controller application as a separate python file and describe how it performs bandwidth calculation.

c) **React to traffic fluctuations**

As a final step, you want to re-route HAS traffic via VyOS-3 whenever the bandwidth of the iperf flow exceeds the threshold. However, since routing via VyOS-3 is associated with higher costs, HAS traffic shall be routed via VyOS-2 whenever the bandwidth of the iperf traffic is below the threshold.

Implement the remaining functionality and include the final python module in your submission.

Proud of your bandwidth-aware routing mechanism, you want to show off to management and highlight the bandwidth savings on the VyOS-3 links that you achieved in comparison to the original approach which put all HAS traffic on the VyOS-3 route. Use the bandwidth monitoring data to create a time series plot over a duration of at least 15 minutes with two curves: one curve for the amount of traffic (in Mbps) that is routed via VyOS-2 and one for the VyOS-3 route.

Include the plot in your report and discuss the results. Furthermore, discuss how changes to the bandwidth-threshold and monitoring interval would affect the behavior of your controller application.

Is it possible to implement the same functionality without using SDN, e.g., using the SNMP / NetFlow setup from the first lab? How would you design such a system? What would be the resulting trade-offs and pitfalls?

Task 1.3 Reflection

a) **Layer violation**

By replacing a router (Layer 3) with a switch (layer 2) you provoke a layer violation. Reflect again on why this is a problem and how you solve this issue in the assignment. Even with the layer violation, we manages to send traffic through the network as intended. Why is this not done regularly when it works?

Submission details

Please submit an archive (.tar.gz or .zip) which contains all files you want to submit. Please have your group number in the file name and the directory name. Submission channel will be shared during the lectures. A report (one single PDF file, named worksheet(num)-group(num).pdf) containing the following elements is mandatory:

- Your group number on the first page.
- For each question, the written answers with the **relevant** portions of output from all commands in text format. No screenshots of terminal windows are accepted.
- For each question, all **changed** parts in the configuration of routers and hosts (differences to the default config).
- **Never** include the full verbatim switch or router configuration in the pdf report.
- Do not include the full SDN-controller code in the report, but include the files in the zip.
- For all questions, state your assumptions, report what you did, describe what you observed, and explain your conclusions.

Additionally, please include the source code of your implemented controller modules in the archive. Each subtask starting from Task 1.1(b) should produce one functional controller module. We can only grade what we find in your submission and what we understand. Please state your assumptions and observations as clearly as possible.

Due date: Friday, 18th of September 2026, 11:59 PM