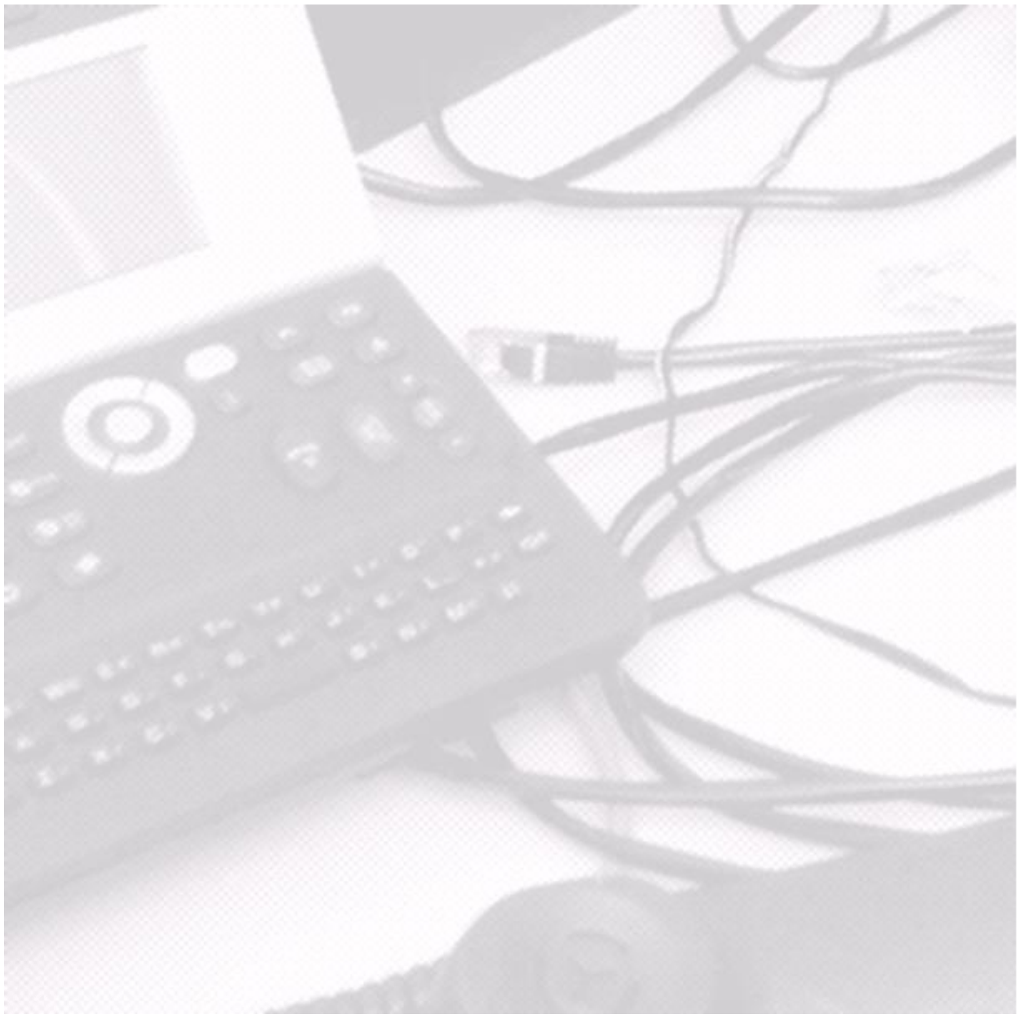
ZSUT

Zakład Sieci i Usług Teleinformatycznych

EINES Project Description



Project 1: QoS routing with SDN

Prepared by: dr inż. Dariusz Bursztynowski

ZSUT. Zakład Sieci i Usług Teleinformatycznych
Instytut Telekomunikacji
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

April 2022

Contents

1. General

The goal of this project is to gain basic understanding of the role of SDN controller in providing northbound services based on the capabilities offered by the southbound interface.

The project relies on the tools used and the skills acquired during Lab 1. Actually, the project consists in adopting Python scripts of the controllers used in Lab 1 by combining them to implement a new controller. The new controller is expected to dynamically select the routes for traffic flows in accordance with requirements specified for a given flow and based on the estimated link delays in the network. The requirements are provided to the controller from “application” generated command that models an “intent” sent to the controller over the northbound interface.

Students are expected to use the virtual machine from Lab 1. **Controller scripts (delay/routing) known form Lab 1 have been updated by providing more detailed comments and their new version is available for download** in the project repository indicated by the instructor.

The following links can be useful during the work on the project:

- OpenFlow specifications
    - https://www.opennetworking.org/software-defined-standards/specifications/
- OpenVSwitch descriptions
    - http://docs.openvswitch.org/en/latest/tutorials/
- Description of ovs-controller
    - http://manpages.ubuntu.com/manpages/trusty/man8/ovs-controller.8.html
- POX controller documentation
    - https://noxrepo.github.io/pox-doc/html/
    - http://intronetworks.cs.luc.edu/auxiliary_files/mininet/poxwiki.pdf

2. Project reporting and evaluation

It is necessary to prepare a report documenting the project. Each project team will be graded based on the report and a demo presented to the instructor during a dedicated meeting.

Mandatory contents of the report is as follows:

- description of project items as detailed in the main part of this instruction
- scripts with a short usage description necessary to run a demo (alternatively, a link in the report to a git repo containing the scripts and usage descriptions)
- summary of the project (not only what has been done, but also what has been learnt, what other topics would be interesting but were not covered by the project, etc.).

3. Project definition

3.1. Introduction

In this project we learn how QoS routing in OpenFlow enabled network can be configured and changed in real time from a central control point according to the requirements expressed by a higher-level "intent". This document mainly covers the basic requirements for a project outcome, leaving many details of the implementation to the students.

3.2. Network operation

We assume a network topology already known from Lab 1 and depicted in Figure 1.

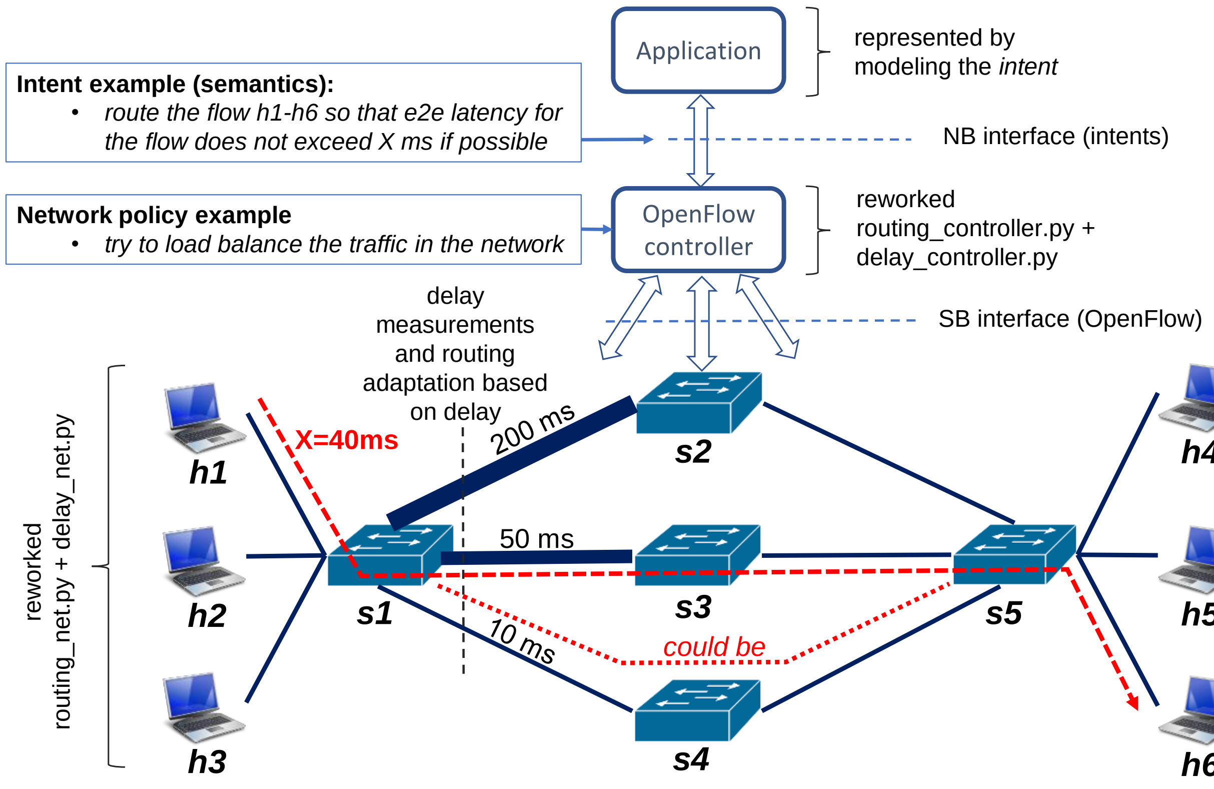

Application
represented by modeling the *intent*
Intent example (semantics):
• *route the flow h1-h6 so that e2e latency for the flow does not exceed X ms if possible*
NB interface (intents)
Network policy example
• *try to load balance the traffic in the network*
OpenFlow controller
reworked routing_controller.py + delay_controller.py
SB interface (OpenFlow)
delay measurements and routing adaptation based on delay
reworked routing_net.py + delay_net.py
X=40ms
200 ms
50 ms
10 ms
could be
h1
h2
h3
h4
h5
h6
s1
s2
s3
s4
s5

Figure 1 Network topology and operation assumed for the project.

In the network considered, **the Application can request the controller to set up a flow for a given host pair** (traffic relation) so that the end-to-end delay for such a relation does not exceed a value defined in the request. In our context, such a request will be referred to as ***intent*** presented (delivered|) to the controller using the NB interface. The example intent shown in Figure 1 specifies the required latency for the relation h1-h6 should be below 40 ms. For sake of simplicity, in the project **we assume zero delays on links other than s1-s2/3/4**; this results in e2e delay being determined by the latency of a single link (no shortest path calculation is required). As a result, the controller sets the route (flow in the flow table) for this relation in switch S1 using link s1-s3 which satisfies the 40ms requirement. Notice that a route through the link s1-s4 would also work in this case. However, the final choice can be subject to other constraints/network policies (like load balancing) or "magic" tie-breaking rules internal to the controller.

We assume that the **traffic (flows) for unspecified relations can take any route**, but when assigning such routes the controller should always account for the constraints imposed by (internal) network policies. In Figure 1, an example network policy demands that the traffic is load balanced in the network. The exact meaning of "load balancing" (whether we load balance byte or packets transferred during some time, or concurrent flows active in network fragments, and what is the scope of the balancing procedure – link/node/network, etc.), and the details of respective algorithm including tie-breaking rules are assumed to be specific to the controller.

In our project, students are expected do define their load balancing rule. Some examples to select from are as follows:

- equalize the number of active flows on the links in the link set s1-s2/3/4 (more precisely, apply min-max rule to the number of flows on these links)
- equalize the number of bytes sent through the links in the link set s1-s2/3/4 within some time interval (different averaging rules can be applied, e.g., fixed window, sliding window, etc.)
- variations of the above, but on the packet level and applied to the nodes not links, etc.

The first of the above criteria seems to be most suitable for our project as it is directly calculable, while the other two examples require gathering statistics from the switches and processing them to obtain averages. As also seen, in our project we assume that the operational scope of load balancing is limited to a subset of links. Although it is not entirely global, still provides sufficient insight into the idea of using policies in network control.

The role of the controller in our scenario is to monitor the e2e delay for each intent-defined relation during the lifetime of the flow (e.g., for h1-h6 as in the figure) and dynamically change routing for this flow in switch s1 so that the maximum latency requirement is satisfied. At the same time the controller should monitor link load parameters and reroute flows according to the load balancing network policy[1]. Resolution of tie-breaking rules when appropriate is also up to the students.

3.3. Project scope and grading (or what should be done and how much it values)

The task

The task consists in designing and implementing:

- a simple intent conforming to the description from section 3.2.
- a controller able to consume intents and react to them in a way described in section 3.2
- choosing one out of the two following design options:
  - ***basic***: it is sufficient that only one flow at a time can be subject to intent-based treatment (i.e., be monitored and rerouted when needed according to the specification given in the intent)
  - ***extended***: multiple flows can be subject to intent-based treatment in parallel
- under the assumption that the scope of delay monitoring and routing operation is limited only to the links outgoing from switch s1 to switches s2, s3, s4 and flows in the forward (from left to right) direction (more details on that in section 4)
- and **documenting major steps** of your work including basic tests of the implementation.

[1] Notice that although theoretically all flows can be rerouted in one cycle, but this may be not realistic in operational network, and other policies might exist to keep the number of rerouted flows to a minimum. For sake of simplicity, we do not consider such constraints in our project.

Grading

The grade for a solution compliant with **option *basic* is 5** (out of 5), while a solution for **option *extended* can be granted 20% bonus**.

4. Implementation guidelines

- We assume that the scope of delay monitoring and routing operation can be limited only to the links between switch s1 and switches s2, s3, s4, and only for the traffic direction FROM hosts h1/h2/h3 TO hosts h4/h5/h6. This should simplify your work.
- We reuse the virtual machine and the scripts known from Lab 1. However, it is recommended to update the scripts **`delay_controller.py`** and **`routing_controller.py`** from the project repository indicated by the instructor. The scripts now contain more detailed comments that may ease the work on code adaptation for the needs of the project.
- The topology of the network can (should be) created using Mininet script written in Python named **`routing_net.py`**. Script **`delay_net.py`** should be used as a blueprint for implementing the capability that allows to vary the delay on the links. This can be done either by hardcoding a loop with delay modifications in **`routing_net.py`** or manually from the Mininet command line. The decision as to which option to use is left up to the students, although the first option is recommended as it has already been tested in Lab 1.
- The controller can be implemented by extending the script **`routing_controller.py`** with intent handling capability, delay measurement functionality, and routing calculation (for both intent-defined and for undefined flows to load balance the traffic[2]). Script **`delay_controller.py`** can be used as a blueprint for organising delay measurements in the new controller. Reminder again: use the new version of both scripts available in the project repo.
- The form of implementation of the intent capability (i.e., how the intent is triggered, formed and injected into the core controller logic) is left to students' decision. From the point of view of the goals of the project any implementation will be valid provided the intent contains all the information needed. Therefore even hardcoding intent creation in the controller script (e.g., on timer expiry) is acceptable.
- The use of timers in Mininet and POX can be studied based on the examples found in delay_net.py and delay_controller.py, respectively.

[2] As already mentioned, load balancing in our example network is very simple as it relates to a set of links outgoing from a single node (s1).