Technical University of Munich, Chair of Communication Networks
Prof. Dr.-Ing. W. Kellerer, Răzvan-Mihai Ursu, Polina Kutsevol, Alba Jano

Analysis, Modeling and Simulation of Communication Networks
SS 2025

Programming Assignment 7

Submission Deadline: 8. July 2025

7 ns-3 Network Simulator Part 2 (70 Points)

In this tutorial, you will build and run basic NS-3 simulations with two routing schemes: distributed and centralized. The routing protocols that you will evaluate are Routing Information Protocol (RIP) and Dynamic Global Routing (DGR). The goals of this tutorial are:

1. Teach you the difference between distributed and centralized routing schemes.
2. Teach you how to implement configurable simulations.
3. Teach you to collect flow statistics with NS-3.

7.1 Deliverables

You must submit your NS-3 code and a Jupyter notebook with plots that answer the indicated questions. Each plot must be described (full sentences) and interpreted. Submit your code in a single archive (ZIP, TAR, etc). Specifically, the archive should contain:

- C++ files (`tutrip.cc`, `tutdgr.cc`)
- **All** generated output files in separate folders located in a directory called data (`data/rip/`, `data/dgr/`)
- Jupyter notebook with plots, code, and answers.

7.2 Setup

One of the goals of this tutorial is to teach you how to use configuration files to quickly evaluate different set-ups without having to change the actual code. For this purpose, this tutorial relies on JSON. The JSON format is a popular and small text-based serialization format and is used, e.g., by Jupyter Notebook to store configurations. For your convenience, the containerized NS-3 already has the `jsoncpp-dev` library installed.

If you do not have a running setup of the containerized NS-3, please follow the setup instructions from Programming Assignment 7. If you have a running container, please first stop it by running:

```
docker stop ns3; docker rm ns3
```

Next, download from Moodle the archive of the skeleton code of the current task `NS3_part2_skeleton` and extract it.

Additionally, create a folder named `output` You should have a file structure as the one depicted below:

```
assignment_7/
|-- NS3_part2_skeleton/
| |-- CMakeLists.txt
| |-- topology.json
| |-- tutdgr.cpp
| \-- tutrip.cpp
|-- output/
```

While in the `assignment_7` folder, run the NS-3 Docker Image in the daemon mode `-d`, mounting the skeleton folder to the `/usr/ns3/ns-3.42/scratch` and the `output` folder to `/usr/ns3/ns-3.42/output`, by running the command:

```
docker run -d \
-v ${PWD}/NS3_part2_skeleton:/usr/ns3/ns-3.42/scratch \
-v ${PWD}/output:/usr/ns3/ns-3.42/output \
--name ns3 \
gitlab.lrz.de:5005/lkn_teaching/ams/ns3-docker/ns3-docker:main \
sleep infinity
```

After the container starts, you should be able to open a `bash` Terminal in your container with:

```
docker exec -it ns3 bash
```

Developing the simulations and running the tests

Open the `assignment_7` in a code editor and solve tasks 7.3 - 7.5. Because the skeleton folder is mounted as a Volume to the container, the `*.cpp` files in the local skeleton folder on your machine are synchronized with those in the Docker container in the `/usr/ns3/ns-3.42/scratch` Docker container.

While in a Terminal in the container, you can build and run target by running the command

```
./ns3 run '<target> <cli-args>'
```

Design your code so that the data files are output in the folder `/usr/ns3/ns-3.42/output` of the Docker container. Because of the mounted Volume, you will observe these files also in the `output` folder on your machine. Process the obtained data files on your machine with suitable Jupyter Notebooks in order to conduct the Simulation Studies.

Cleaning up

To exit the `bash` Terminal in you container, run in the container

```
exit
```

To stop the container, run on your host

```
docker stop ns3
```

To remove the container, run on your host

```
docker rm ns3
```

To delete the pulled image, run on your host

```
docker rmi gitlab.lrz.de:5005/lkn_teaching/ams/ns3-docker/ns3-docker:main
```

7.3 Dynamic Global Routing

25 Points

In this task you have to use `Dynamic Global Routing` as routing protocol. This routing protocol corresponds to the `Ipv4GlobalRouting` class. The routing tables of the router nodes should be configured with this protocol. Ensure that the routing protocol *responds to interface events*.

The goal of this exercise is to read in the provided JSON configuration file and complete the skeleton, such that the simulation defined in the JSON file is correctly executed. Specifically:

- Read the JSON file.[1]
- Configure nodes, links, devices, IP-addressing and routing correspondingly.
- Configure a flow monitor to get information about the flow.
- Trace the RTT of the Ping application using NS-3's tracing system. Collect the RTTs together with the corresponding simulation time.

Once your simulation runs successfully, parse the output of the flow-monitor and the ping statistics and answer the following questions:

1. How many flows are captured in the flow-monitor? Why did you observe those flows and to what do they correspond?
2. Plot the number of dropped packets vs. the number of received packets. When do those packet drops occur? Is the amount of dropped packets reasonable given how DGR works?
3. Compute the average per-packet latency.
4. Compute the average per-packet jitter. Where does the jitter come from?

[1] Should you be unable to parse the JSON file, you can also hardcode the configuration in your code, with a penalty of 5 points.

5. Plot the RTT measured with the Ping application over time. Explain the behavior.

Describe and interpret the plots as well as answer the associated questions.

7.4 RIP

30 Points

In this task you have to use `Routing Interface Protocol` as routing protocol. This routing protocol corresponds to the `RIP` class in NS-3. The routing tables of the router nodes should be configured with this protocol.

The goal of this exercise is to read in the provided JSON configuration file and complete the skeleton, such that the simulation defined in the JSON file is correctly executed. Specifically:

- Read the JSON file.
- Configure nodes, links, devices, IP-addressing and routing correspondingly.
- Configure a flow monitor to get information about the flow.
- Trace the RTT of the Ping application using NS-3's tracing system. Collect the RTTs together with the corresponding simulation time.
- Write the routing table of each node in the network to a separate file.

In case of `RIP`, you have to distinguish between routers in the backbone and hosts. Routers in the backbone network are configured as `RIP` router. Hosts are configured with static routing, i.e., they are not configured as `RIP` routers.

Once your simulation runs successfully, parse the output of the flow-monitor and the ping statistics and answer the following questions:

1. How many flows are captured in the flow-monitor? Why did you observe those flows and to what do they correspond?
2. Plot the number of dropped packets vs. the number of received packets. When do those packet drops occur? Is the amount of packets reasonable given how `RIP` works?
3. Parse the dumped routing tables and identify the times, the table changed. Show the corresponding entries in the file. Why do the tables change in the way they do?
4. Compute the average per-packet latency.
5. Compute the average per-packet jitter. Where does the jitter come from?
6. Plot the RTT measured with the Ping application over time. Explain the behavior.

Describe and interpret the plots as well as answer the associated questions.

7.5 DGR vs RIP

15 Points

- Compare the number of lost packets of `DGR` and `RIP` with each other and explain your observation.

- Compare the average per-packet jitter of `DGR` and `RIP` with each other and explain your observation.

7.6 Tips

If you have difficulties getting started check out the following resources on NS3:

- For the behavior of the RTT in case of RIP you might want to take a look at the routing tables.
- Example for Dynamic Global Routing[2]
- Example for RIP[3]
- How to run scripts in NS-3[4]

Total: 70 Points

[2] `https://www.nsnam.org/doxygen/dynamic-global-routing_8cc_source.html`
[3] `https://www.nsnam.org/doxygen/rip-simple-network_8cc_source.html`
[4] `https://www.nsnam.org/support/faq/running-scripts/`