This is a README file. In this file, we will detail the file structure, how to compile and run the code and check the output.

1 File structure

The main files to be used are the following:

1. `main.cpp` This is the main function of the AXI4-NoC by which it sends states to the RL agent and receives actions from it.
2. `interfaceCPP.cpp` and `interfacePy.py` form the data transfer interface.
3. `RLtrain_new.py` is used to train the SAC agent.
4. `RLevaluation.py` load the trained SAC agent to perform admission control in NoC.
5. `NoC_SAC_1.zip` stores the weight parameters of the SAC model.
6. `./RecordFiles/OnlyURStransLevel_delay_20230118.py`:This written by Yizhi and it's capable of counting a lot of data. Here we've mainly used the programme to calculate the average end-to-end latency.
7. `./RecordFiles/60000_2025_epoch_0_URS_packetdelay.txt` It stored the time data from the NoC operation without admission control.
   `./RecordFiles/60000_2025_epoch_1_URS_packetdelay.txt` It stored the time data from the NoC operation with the SAC agent.

2 Running and results

This process is a bit complicated by the fact that we have not written an automated execution script.

1. You need to install a C++ IDE and compile `main.cpp` into an executable file.
2. You need to configure Python 3.8 or above and install the relevant libraries, mainly the Stable-Baselines3 library.You can try this command `pip install stable-baselines3[extra]`, If the above command do not work, please visit their official website directly.
3. Since we have already trained the model, you can directly use the existing model to perform admission control in the NoC. Specifically, first execute `python3 RLevaluation.py`.When the terminal prints out the "reset ,begin", then excute `mian.cpp` .
4. At this point, you should pay attention to the output of `main.cpp`, when the first time to print the results as shown in Figure 1 , first interrupt `mian.cpp` and then interrupt `RLevaluation.py`. This is because the results of each evaluation are the same, the NoC's operation data has been completely recorded no longer need to evaluate the second time.

5. After recording the NoC operation data, you can execute `./RecordFiles/OnlyURStransLevel_delay_20230118.py` This will result in two histograms recording the distribution of the end-to-end latency of the signals. However, we did not use this two figures in our report. At this point you can also focus on the print message, which outputs some of the parameters we used in the final report, such as Figure 2. If you want to directly observe the average end-to-end latency after performing admission control, don't forget to change this line of code in this program `txtFileName_URS = "60000_2025_epoch_0_URS_packetdelay"` to `txtFileName_URS = "60000_2025_epoch_1_URS_packetdelay"`.

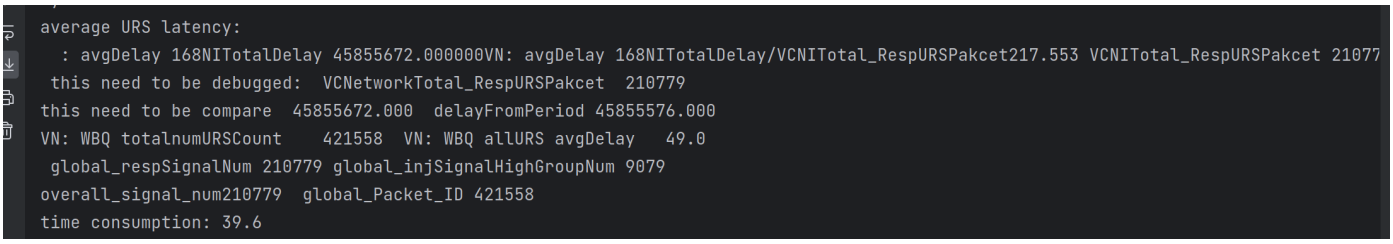

Figure 1: The output of mian.cpp

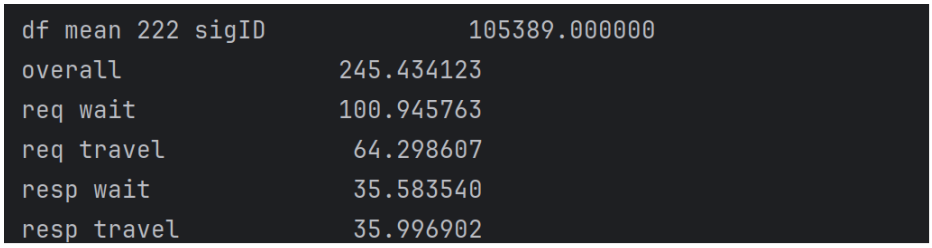

Figure 2: The output of OnlyURStransLevel_delay_20230118.py

Figure 1 shows some important parameters. request signal number=210779,response signal number =210779 and the total packet number= 421558 .All of this parameters show the generated packets are delivered to the corresponding destinations.

In my original code I also included the Python 3.8 directory, the code in GitHub does not include it, and I'm not sure if this affects the proper functioning of the programme.