1. Advanced Network Simulation

1. Create a small 5G network with two base stations and 10 UEs.

In teacher's tutorial, he run this to set a simple 5G network. We key --PrintHelp for getting information of parameters.(Fig. 1) By reading the description, we can set gNbNum to two and ueNumPergNb to ten to satisfy the problem and get what we want.(Fig. 2)

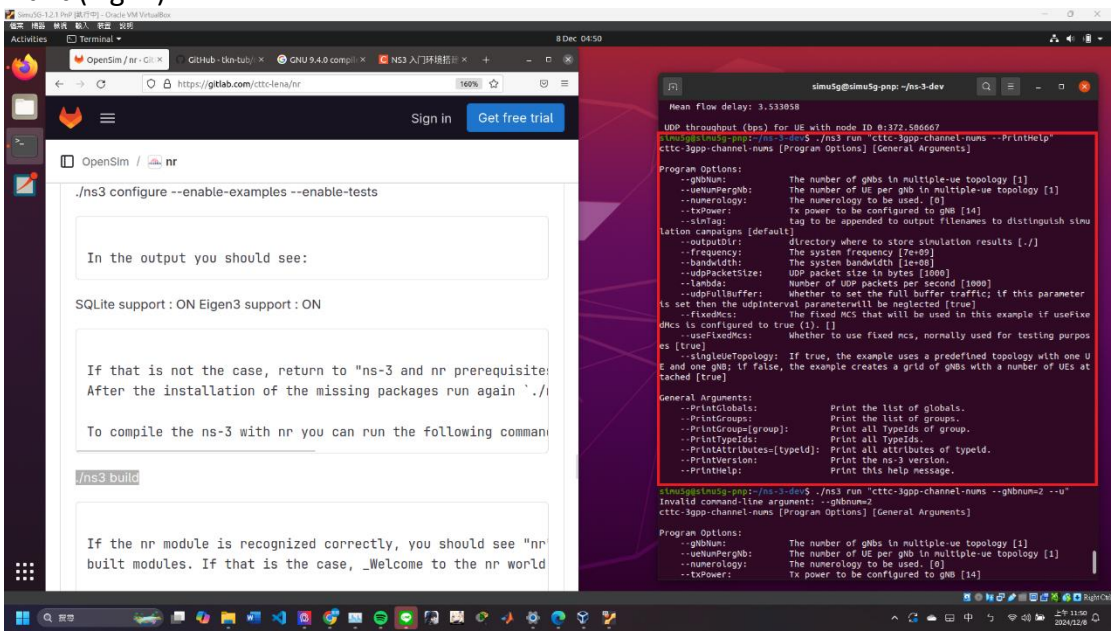
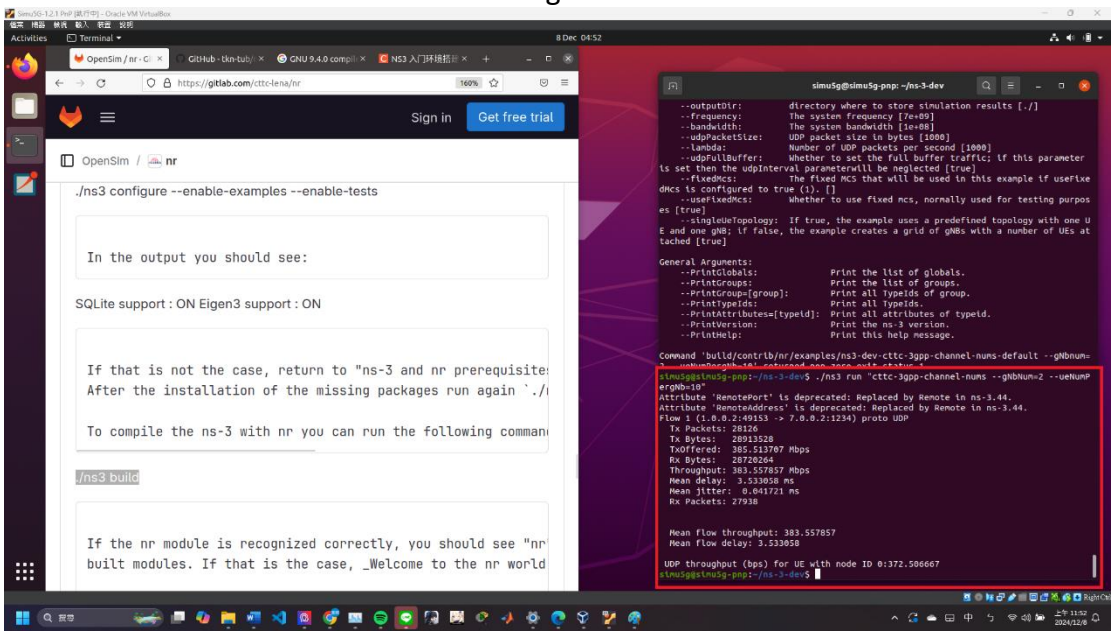


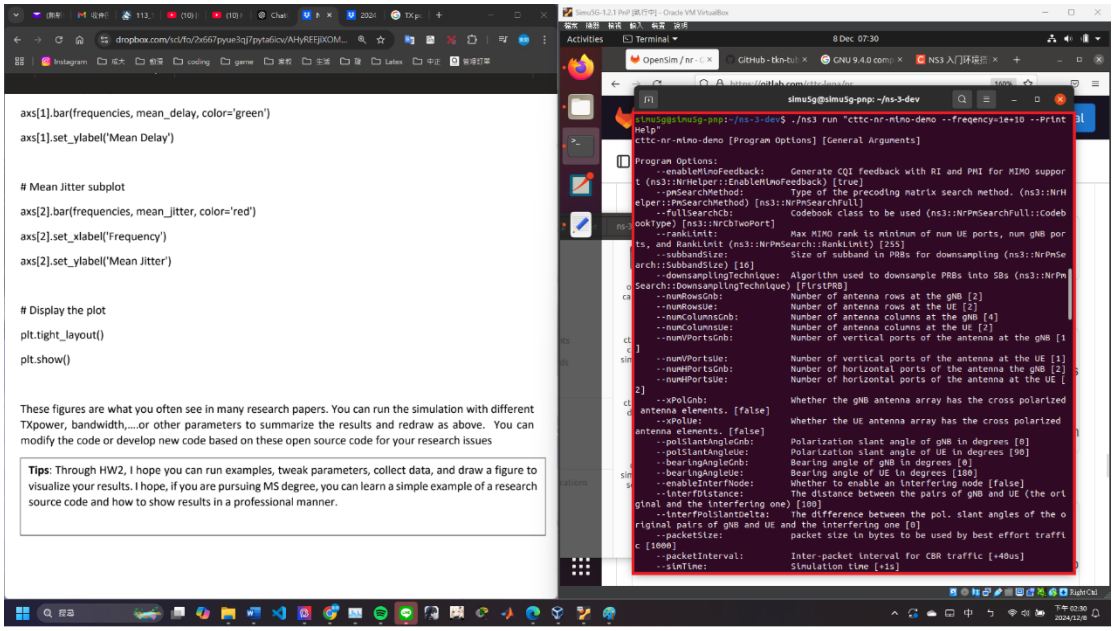
Fig. 1



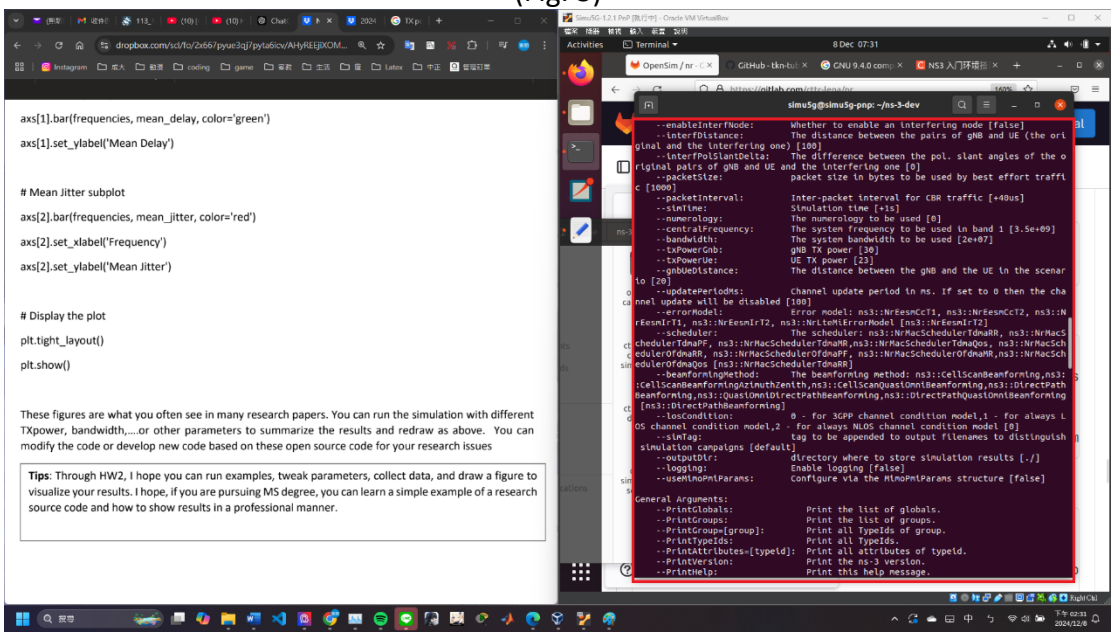
(Fig. 2)

2. Test an application (in examples/test folders) and validate its throughput, delay, jitter metrics.

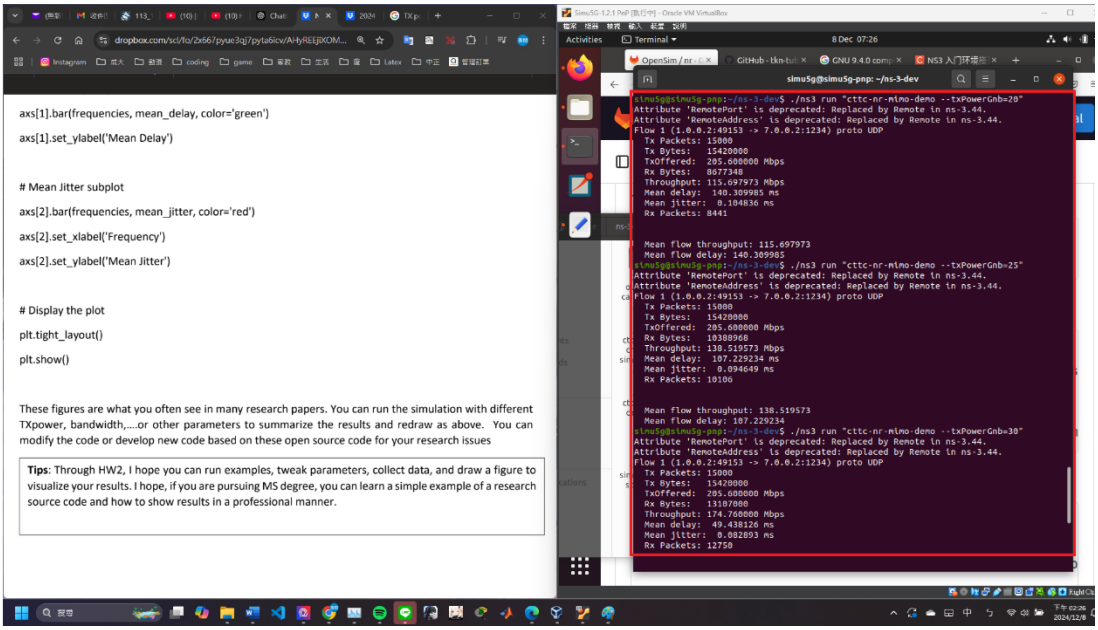
Here we choose cttc-nr-mimo-demo to test. We also key –PrintHelp for getting information of parameters.(Fig. 3,4) By reading the description, we choose txPowerGnb to test what are the different results for different txPowerGnb. We set txPowerGnb to 20, 25, 30, 35, 40 respectively. (Fig. 5~7)


axs[1].bar(frequencies, mean_delay, color='green')
axs[1].set_ylabel('Mean Delay')
Mean Jitter subplot
axs[2].bar(frequencies, mean_jitter, color='red')
axs[2].set_xlabel('Frequency')
axs[2].set_ylabel('Mean Jitter')
# Display the plot
plt.tight_layout()
plt.show()
These figures are what you often see in many research papers. You can run the simulation with different TXpower, bandwidth,....or other parameters to summarize the results and redraw as above. You can modify the code or develop new code based on these open source code for your research issues
Tips: Through HW2, I hope you can run examples, tweak parameters, collect data, and draw a figure to visualize your results. I hope, if you are pursuing MS degree, you can learn a simple example of a research source code and how to show results in a professional manner.
simu5g@simu5g-pnp: ~/ns-3-dev
./ns3 run "cttc-nr-mimo-demo --freqency=1e+10 --PrintHelp"
cttc-nr-mimo-demo [Program Options] [General Arguments]
Program Options:
--enableMimoFeedback: Generate CQI feedback with RI and PMI for MIMO support (ns3::NrHelper::EnableMimoFeedback) [true]
--pmSearchMethod: Type of the precoding matrix search method. (ns3::NrHelper::PmSearchMethod) [ns3::NrPmSearchFull]
--fullSearchCb: Codebook class to be used (ns3::NrPmSearchFull::CodebookType) [ns3::NrCbTwoPort]
--rankLimit: Max MIMO rank is minimum of num UE ports, num gNB ports, and RankLimit (ns3::NrPmSearch::RankLimit) [255]
--subbandSize: Size of subband in PRBs for downsampling (ns3::NrPmSearch::SubbandSize) [16]
--downsamplingTechnique: Algorithm used to downsample PRBs into SBs (ns3::NrPmSearch::DownsamplingTechnique) [FlatPRB]
--numRowsGnb: Number of antenna rows at the gNB [2]
--numRowsUe: Number of antenna rows at the UE [2]
--numColumnsGnb: Number of antenna columns at the gNB [4]
--numColumnsUe: Number of antenna columns at the UE [2]
--numVPortsGnb: Number of vertical ports of the antenna at the gNB [1]
--numVPortsUe: Number of vertical ports of the antenna at the UE [1]
--numHPortsGnb: Number of horizontal ports of the antenna the gNB [2]
--numHPortsUe: Number of horizontal ports of the antenna at the UE [2]
--xPolGnb: Whether the gNB antenna array has the cross polarized antenna elements. [false]
--xPolUe: Whether the UE antenna array has the cross polarized antenna elements. [false]
--polSlantAngleGnb: Polarization slant angle of gNB in degrees [0]
--polSlantAngleUe: Polarization slant angle of UE in degrees [90]
--bearingAngleGnb: Bearing angle of gNB in degrees [0]
--bearingAngleUe: Bearing angle of UE in degrees [180]
--enableInterfNode: Whether to enable an interfering node [false]
--interfDistance: The distance between the pairs of gNB and UE (the original and the interfering one) [100]
--interfPolSlantDelta: The difference between the pol. slant angles of the original pairs of gNB and UE and the interfering one [0]
--packetSize: packet size in bytes to be used by best effort traffic [1000]
--packetInterval: Inter-packet interval for CBR traffic [+40us]
--simTime: Simulation time [+1s]

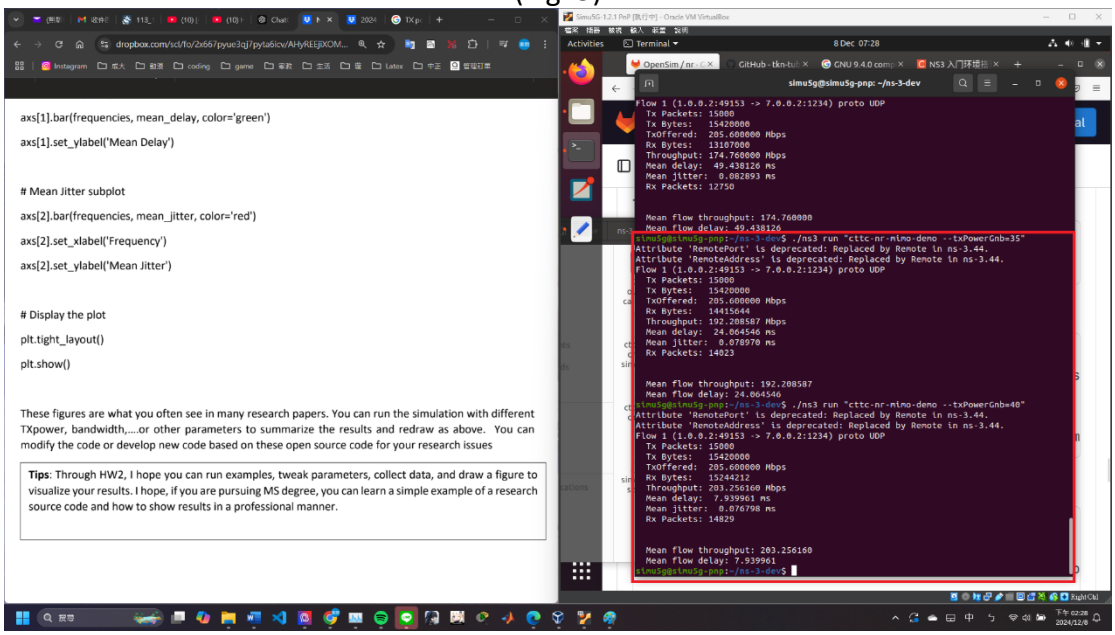
(Fig. 3)


axs[1].bar(frequencies, mean_delay, color='green')
axs[1].set_ylabel('Mean Delay')
Mean Jitter subplot
axs[2].bar(frequencies, mean_jitter, color='red')
axs[2].set_xlabel('Frequency')
axs[2].set_ylabel('Mean Jitter')
# Display the plot
plt.tight_layout()
plt.show()
These figures are what you often see in many research papers. You can run the simulation with different TXpower, bandwidth,....or other parameters to summarize the results and redraw as above. You can modify the code or develop new code based on these open source code for your research issues
Tips: Through HW2, I hope you can run examples, tweak parameters, collect data, and draw a figure to visualize your results. I hope, if you are pursuing MS degree, you can learn a simple example of a research source code and how to show results in a professional manner.
simu5g@simu5g-pnp: ~/ns-3-dev
--enableInterfNode: Whether to enable an interfering node [false]
--interfDistance: The distance between the pairs of gNB and UE (the original and the interfering one) [100]
--interfPolSlantDelta: The difference between the pol. slant angles of the original pairs of gNB and UE and the interfering one [0]
--packetSize: packet size in bytes to be used by best effort traffic [1000]
--packetInterval: Inter-packet interval for CBR traffic [+40us]
--simTime: Simulation time [+1s]
--numerology: The numerology to be used [0]
--centralFrequency: The system frequency to be used in band 1 [3.5e+09]
--bandwidth: The system bandwidth to be used [2e+07]
--txPowerGnb: gNB TX power [30]
--txPowerUe: UE TX power [23]
--gnbUeDistance: The distance between the gNB and the UE in the scenario [20]
--updatePeriodMs: Channel update period in ms. If set to 0 then the channel update will be disabled [100]
--errorModel: Error model: ns3::NrEesmCcT1, ns3::NrEesmCcT2, ns3::NrEesmIrT1, ns3::NrEesmIrT2, ns3::NrLteMiErrorModel [ns3::NrEesmIrT2]
--scheduler: The scheduler: ns3::NrMacSchedulerTdmaRR, ns3::NrMacSchedulerTdmaPF, ns3::NrMacSchedulerTdmaQos, ns3::NrMacSchedulerOfdmaRR, ns3::NrMacSchedulerOfdmaPF, ns3::NrMacSchedulerOfdmaMR,ns3::NrMacSchedulerOfdmaQos [ns3::NrMacSchedulerTdmaRR]
--beamformingMethod: The beamforming method: ns3::CellScanBeamforming,ns3::CellScanBeamformingAzimuthZenith,ns3::CellScanQuasiOmniBeamforming,ns3::DirectPathBeamforming,ns3::QuasiOmniDirectPathBeamforming,ns3::DirectPathQuasiOmniBeamforming [ns3::DirectPathBeamforming]
--losCondition: 0 - for 3GPP channel condition model,1 - for always LOS channel condition model,2 - for always NLOS channel condition model [0]
--simTag: tag to be appended to output filenames to distinguish simulation campaigns [default]
--outputDir: directory where to store simulation results [./]
--logging: Enable logging [false]
--useMimoPmiParams: Configure via the MimoPmiParams structure [false]
General Arguments:
--PrintGlobals: Print the list of globals.
--PrintGroups: Print the list of groups.
--PrintGroup=[group]: Print all TypeIds of group.
--PrintTypeIds: Print all TypeIds.
--PrintAttributes=[typeid]: Print all attributes of typeid.
--PrintVersion: Print the ns-3 version.
--PrintHelp: Print this help message.

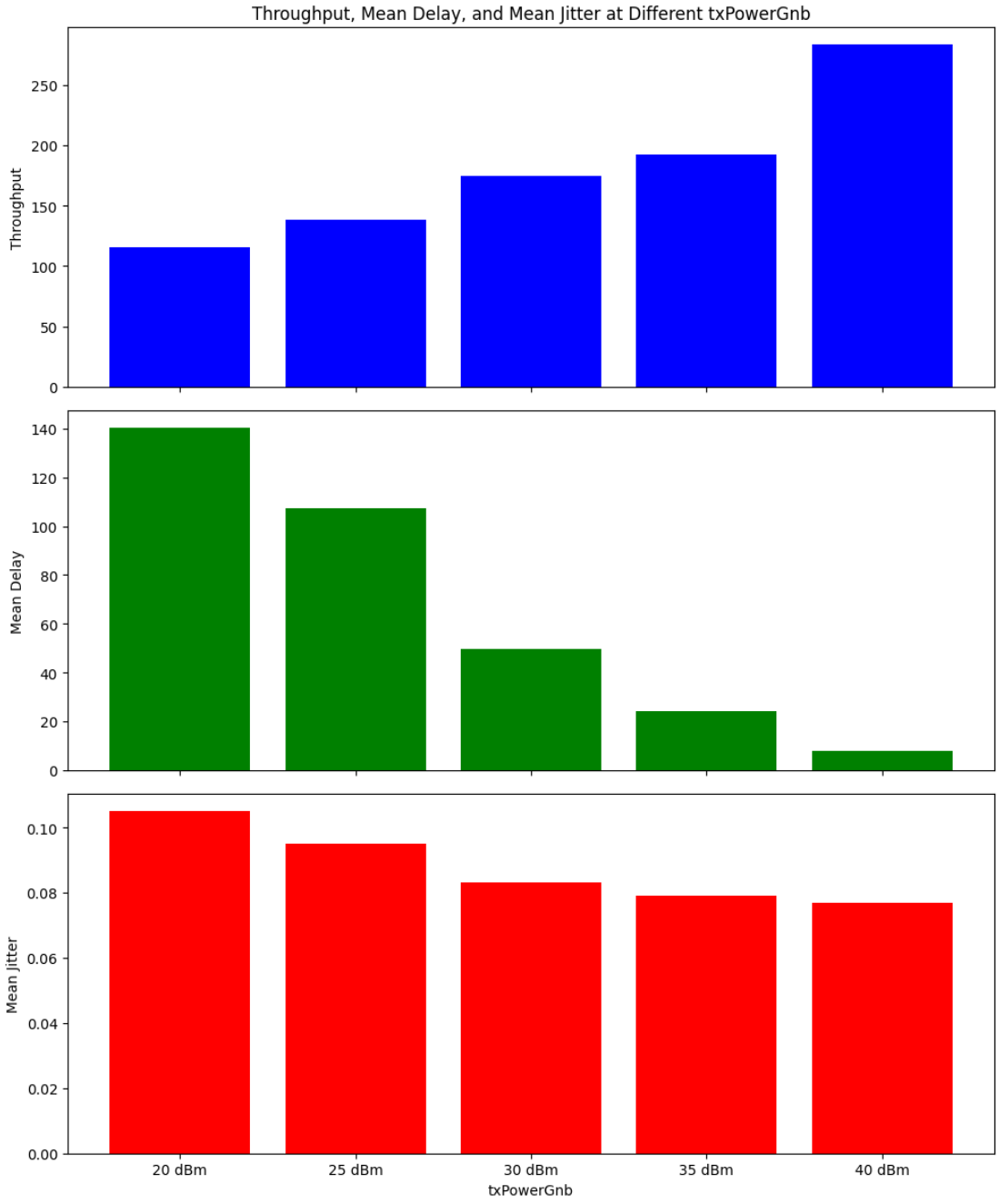
(Fig. 4)


axs[1].bar(frequencies, mean_delay, color='green')
axs[1].set_ylabel('Mean Delay')
Mean Jitter subplot
axs[2].bar(frequencies, mean_jitter, color='red')
axs[2].set_xlabel('Frequency')
axs[2].set_ylabel('Mean Jitter')
# Display the plot
plt.tight_layout()
plt.show()
These figures are what you often see in many research papers. You can run the simulation with different TXpower, bandwidth,....or other parameters to summarize the results and redraw as above. You can modify the code or develop new code based on these open source code for your research issues
Tips: Through HW2, I hope you can run examples, tweak parameters, collect data, and draw a figure to visualize your results. I hope, if you are pursuing MS degree, you can learn a simple example of a research source code and how to show results in a professional manner.
simu5g@simu5g-pnp: ~/ns-3-dev
./ns3 run "cttc-nr-mimo-demo --txPowerGnb=20"
Attribute 'RemotePort' is deprecated: Replaced by Remote in ns-3.44.
Attribute 'RemoteAddress' is deprecated: Replaced by Remote in ns-3.44.
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 8677348
Throughput: 115.697973 Mbps
Mean delay: 140.309985 ms
Mean jitter: 0.104838 ms
Rx Packets: 8441
Mean flow throughput: 115.697973
Mean flow delay: 140.309985
./ns3 run "cttc-nr-mimo-demo --txPowerGnb=25"
Attribute 'RemotePort' is deprecated: Replaced by Remote in ns-3.44.
Attribute 'RemoteAddress' is deprecated: Replaced by Remote in ns-3.44.
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 10388968
Throughput: 138.519573 Mbps
Mean delay: 107.229234 ms
Mean jitter: 0.094649 ms
Rx Packets: 10106
Mean flow throughput: 138.519573
Mean flow delay: 107.229234
./ns3 run "cttc-nr-mimo-demo --txPowerGnb=30"
Attribute 'RemotePort' is deprecated: Replaced by Remote in ns-3.44.
Attribute 'RemoteAddress' is deprecated: Replaced by Remote in ns-3.44.
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 13107000
Throughput: 174.760000 Mbps
Mean delay: 49.438126 ms
Mean jitter: 0.082893 ms
Rx Packets: 12750

(Fig. 5)


axs[1].bar(frequencies, mean_delay, color='green')
axs[1].set_ylabel('Mean Delay')
Mean Jitter subplot
axs[2].bar(frequencies, mean_jitter, color='red')
axs[2].set_xlabel('Frequency')
axs[2].set_ylabel('Mean Jitter')
# Display the plot
plt.tight_layout()
plt.show()
These figures are what you often see in many research papers. You can run the simulation with different TXpower, bandwidth,....or other parameters to summarize the results and redraw as above. You can modify the code or develop new code based on these open source code for your research issues
Tips: Through HW2, I hope you can run examples, tweak parameters, collect data, and draw a figure to visualize your results. I hope, if you are pursuing MS degree, you can learn a simple example of a research source code and how to show results in a professional manner.
simu5g@simu5g-pnp: ~/ns-3-dev
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 13107000
Throughput: 174.760000 Mbps
Mean delay: 49.438126 ms
Mean jitter: 0.082893 ms
Rx Packets: 12750
Mean flow throughput: 174.760000
Mean flow delay: 49.438126
./ns3 run "cttc-nr-mimo-demo --txPowerGnb=35"
Attribute 'RemotePort' is deprecated: Replaced by Remote in ns-3.44.
Attribute 'RemoteAddress' is deprecated: Replaced by Remote in ns-3.44.
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 14415644
Throughput: 192.208587 Mbps
Mean delay: 24.064546 ms
Mean jitter: 0.078970 ms
Rx Packets: 14023
Mean flow throughput: 192.208587
Mean flow delay: 24.064546
./ns3 run "cttc-nr-mimo-demo --txPowerGnb=40"
Attribute 'RemotePort' is deprecated: Replaced by Remote in ns-3.44.
Attribute 'RemoteAddress' is deprecated: Replaced by Remote in ns-3.44.
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
Tx Packets: 15000
Tx Bytes: 15420000
TxOffered: 205.600000 Mbps
Rx Bytes: 15244212
Throughput: 203.256160 Mbps
Mean delay: 7.939961 ms
Mean jitter: 0.076798 ms
Rx Packets: 14829
Mean flow throughput: 203.256160
Mean flow delay: 7.939961

(Fig. 6)

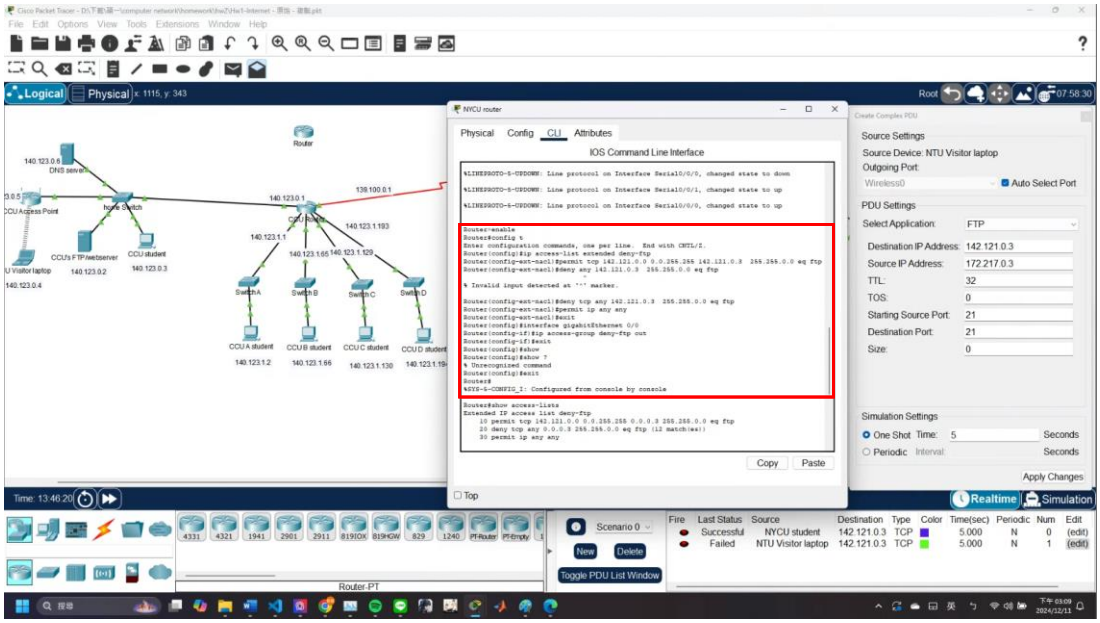


(Fig. 7)

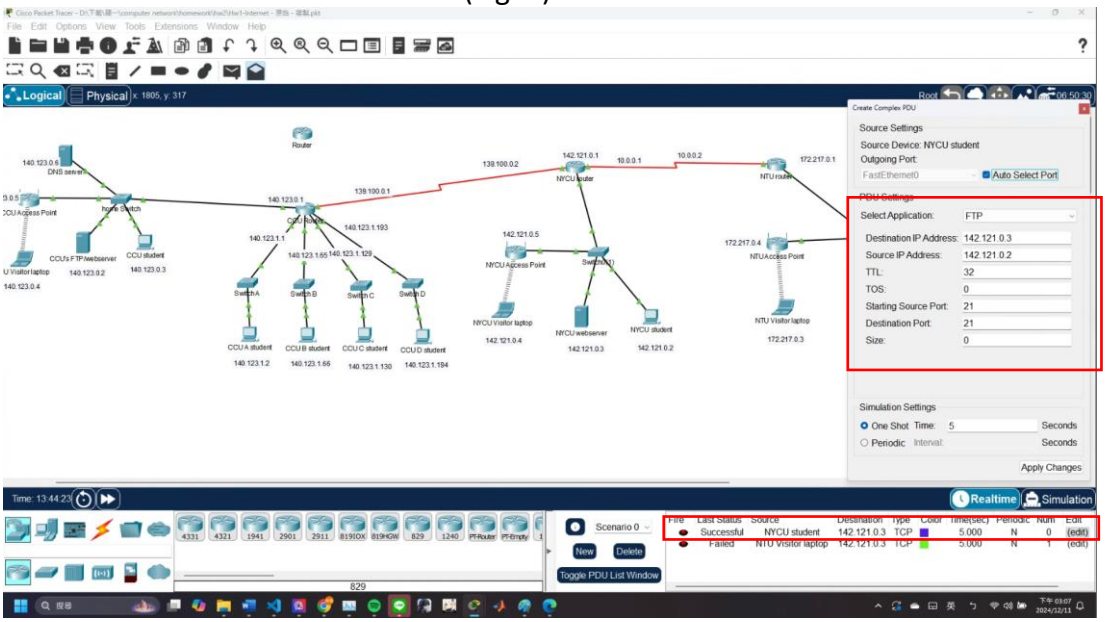
2. Congestion control and network security

3. Firewall configuration

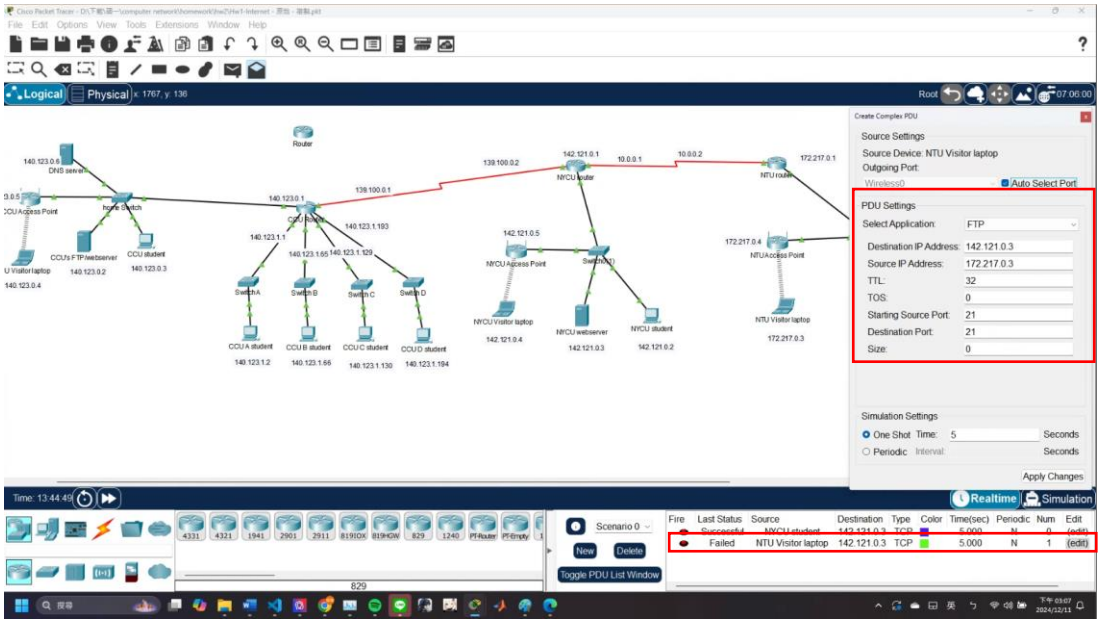
- Deny all FTP connections from outside to access NYCU FTP servers



(Fig. 8) install



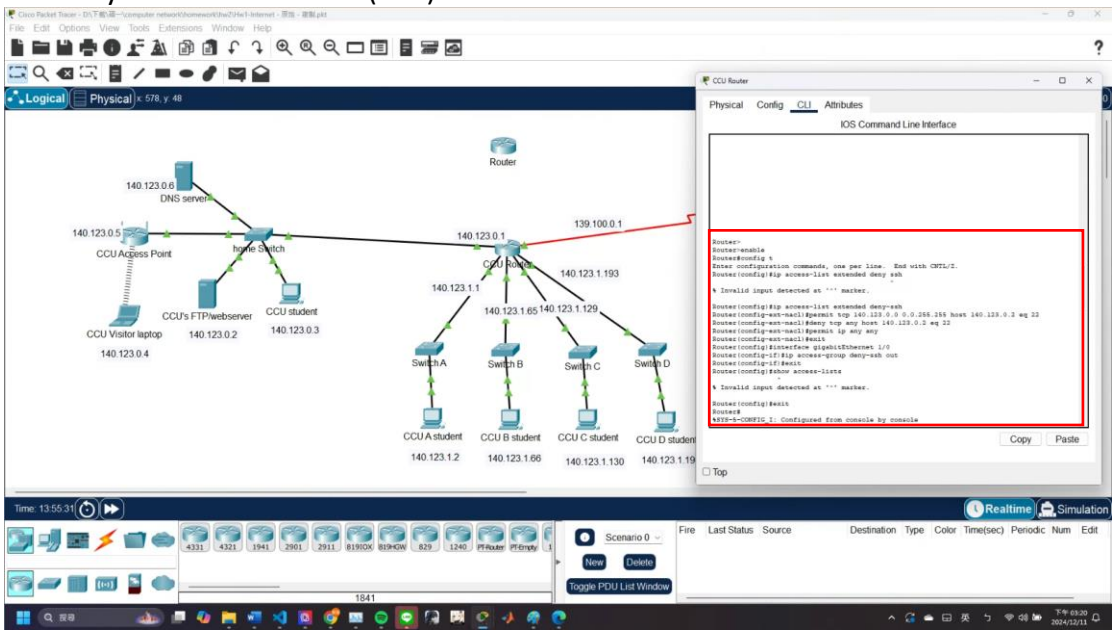
(Fig. 9) test from NYCU student



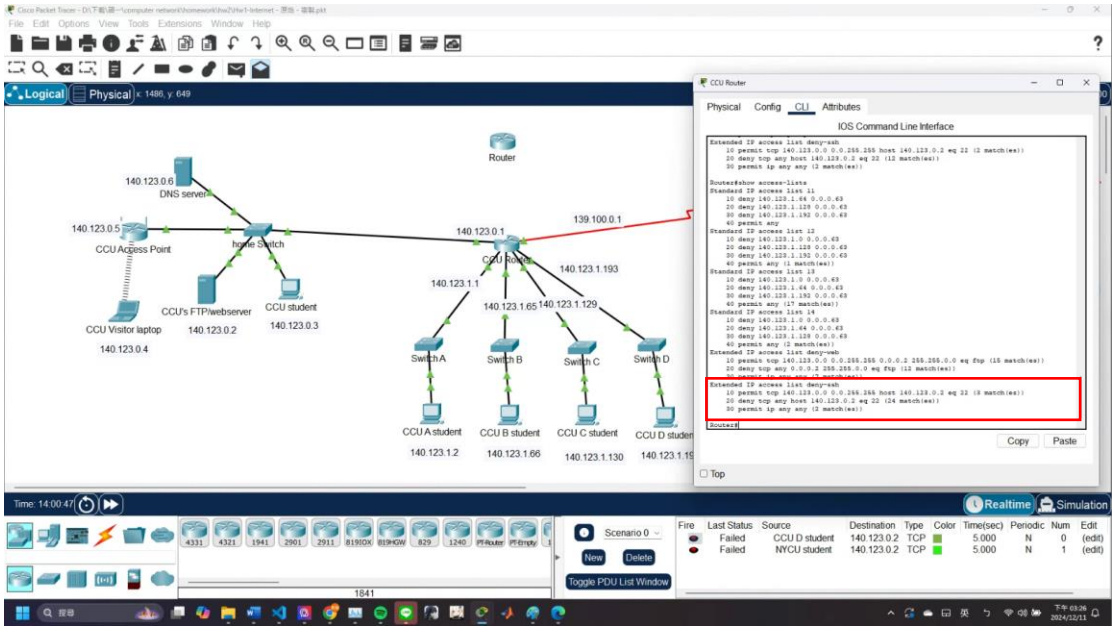
(Fig. 10) test from NTU Visitor laptop

We install(Fig. 8) and test(Fig. 9, 10).

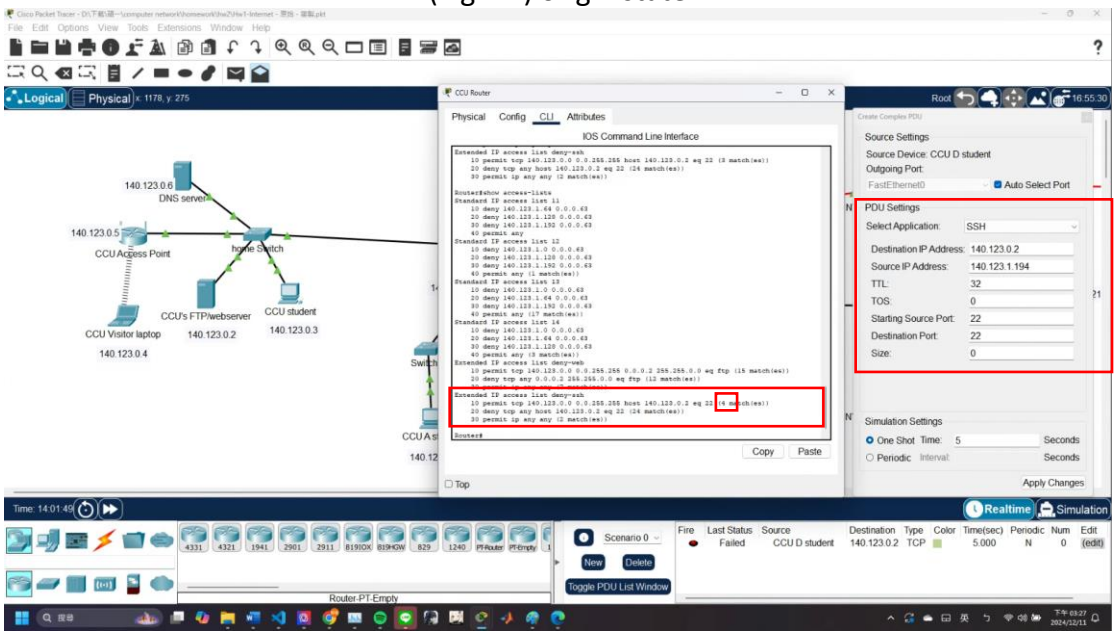
- Deny all remote control (SSH) connections from outside to access CCU server



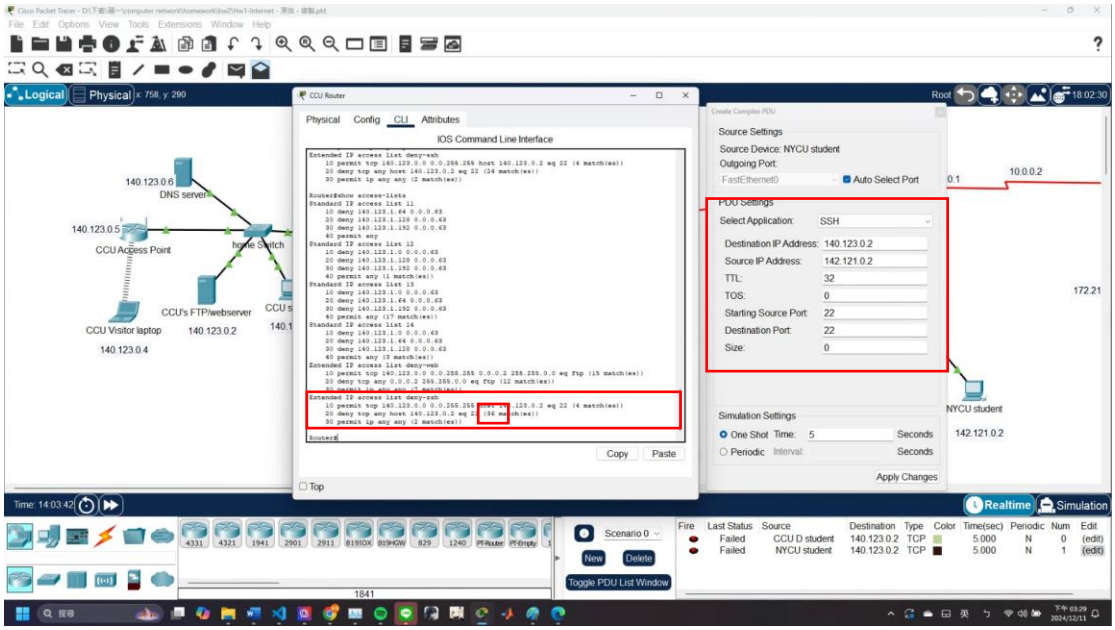
(Fig. 11) install



(Fig. 12) origin state



(Fig. 13) test from CCU D student



(Fig. 14) test from NYCU student

We install(Fig. 11) and test(Fig. 12~14). Last status is failed because ccu server can't be ssh server, it can't response to the request. In the test screenshots we can see the ACL can actually permit and deny!