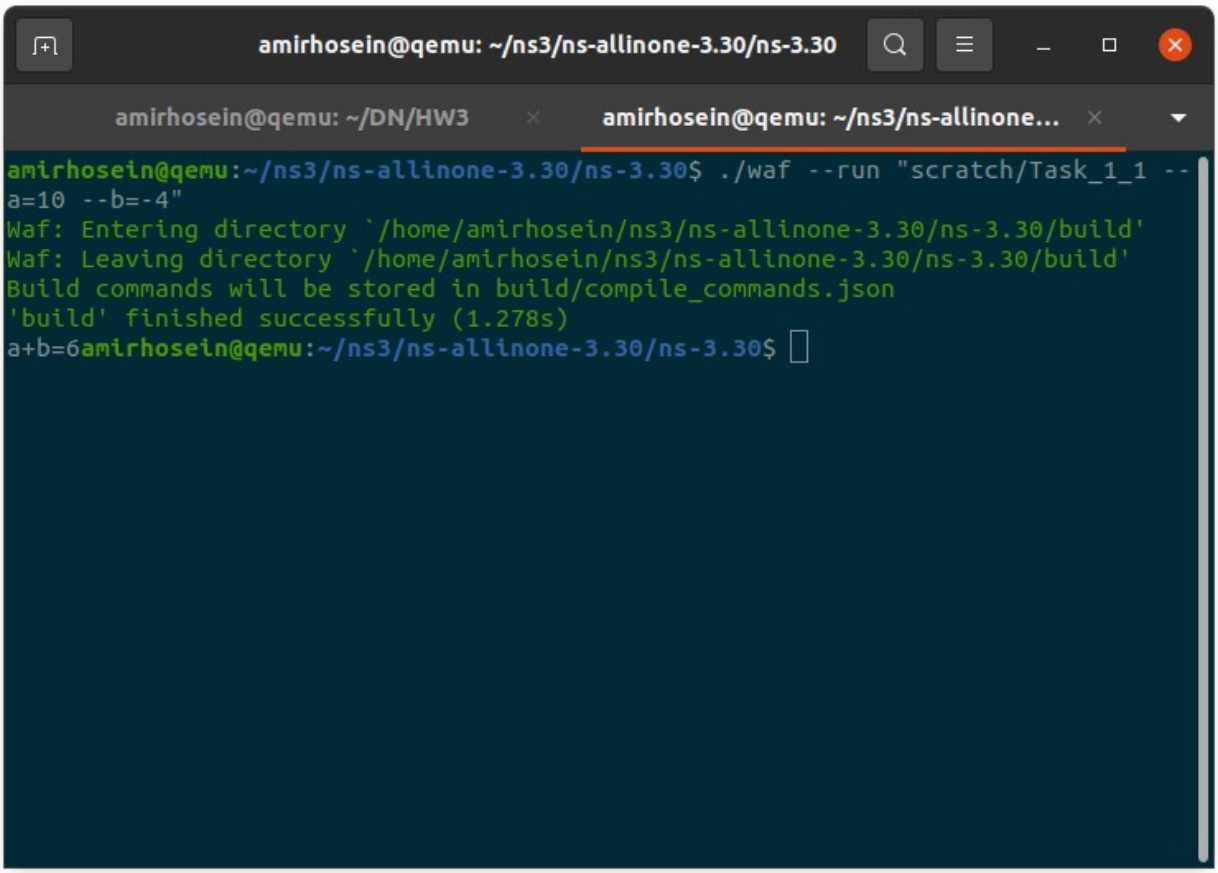
Intro to NS3 : (part 0)

Output of the code is shown below :



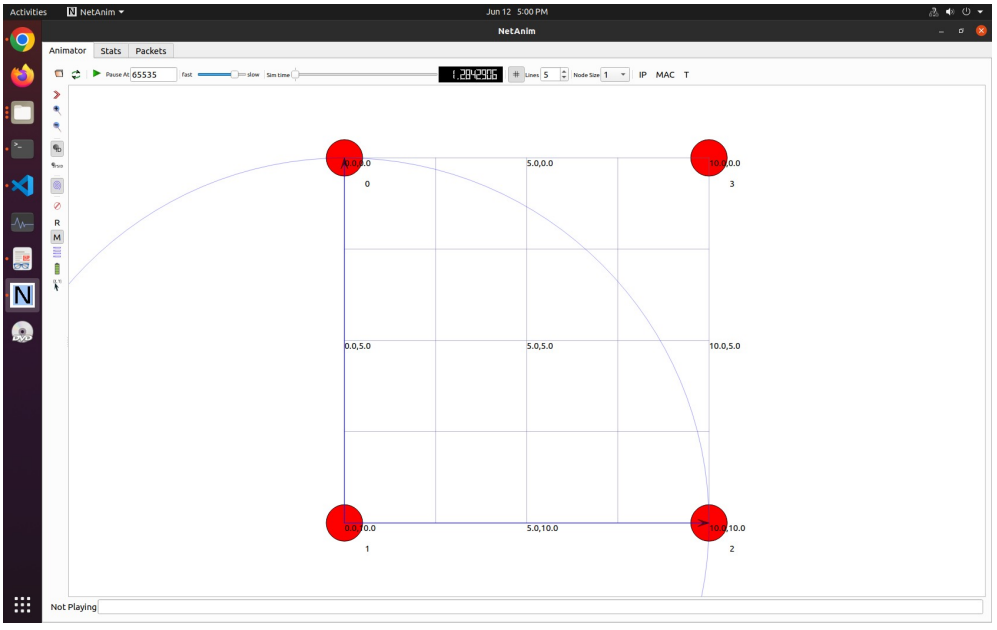
802.11 Simulation :

a) Result is shown below :

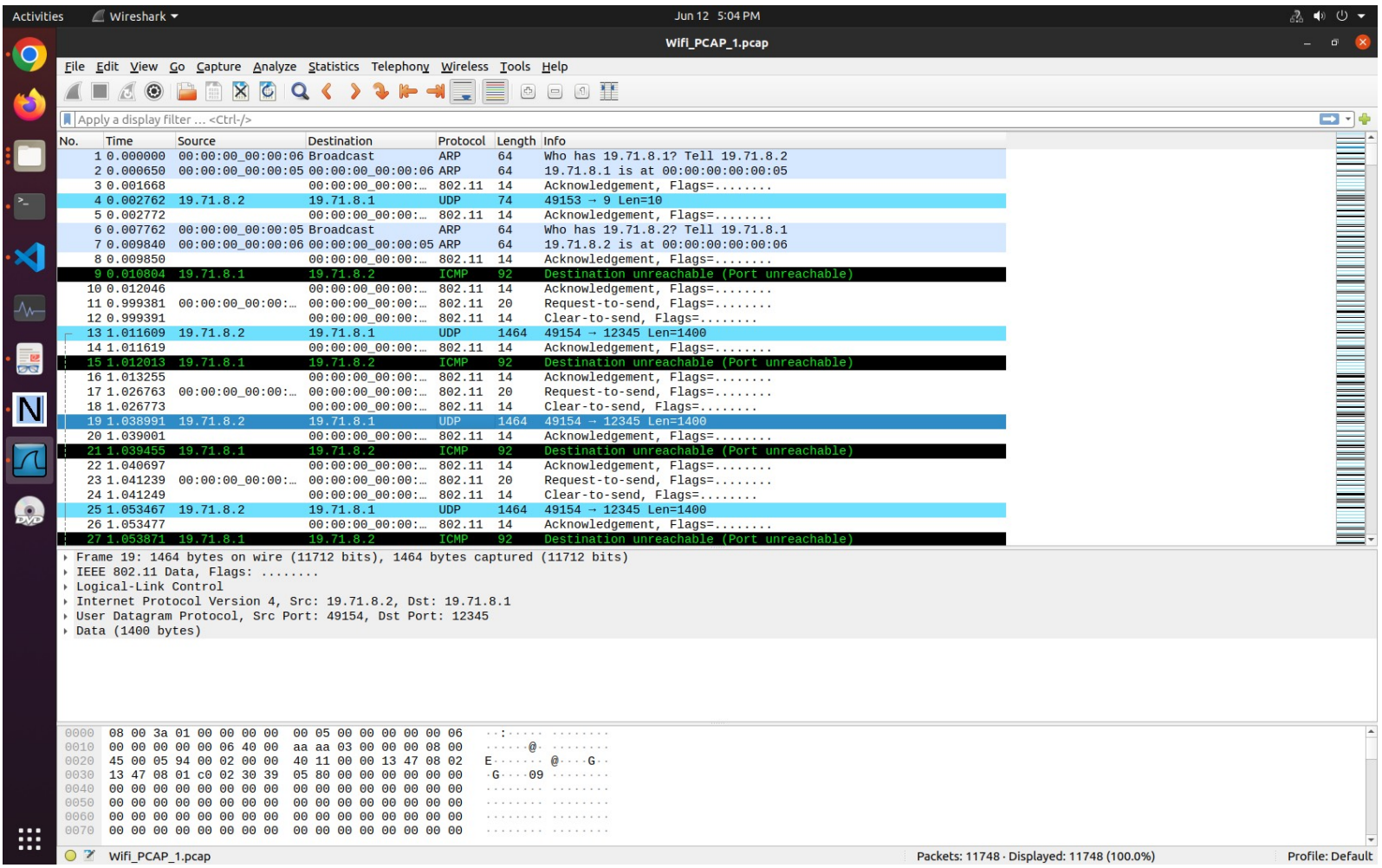
```
amirhosein@qemu:~/ns3/ns-allinone-3.30/ns-3.30$ ./waf --run "scratch/wifi-hidden-terminal"
Waf: Entering directory `/home/amirhosein/ns3/ns-allinone-3.30/ns-3.30/build'
Waf: Leaving directory `/home/amirhosein/ns3/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.293s)
Hidden station experiment with RTS/CTS disabled:
Flow 1 (19.71.8.2 -> 19.71.8.1)
  Tx Packets: 2410
  Tx Bytes:   3441480
  TxOffered:  3.05909 Mbps
  Rx Packets: 2297
  Rx Bytes:   3280116
  Throughput: 2.91566 Mbps
Flow 2 (19.71.8.3 -> 19.71.8.4)
  Tx Packets: 2410
  Tx Bytes:   3441480
  TxOffered:  3.05909 Mbps
  Rx Packets: 2323
  Rx Bytes:   3317244
  Throughput: 2.94866 Mbps
------------------------------------------------
Hidden station experiment with RTS/CTS enabled:
Flow 1 (19.71.8.2 -> 19.71.8.1)
  Tx Packets: 2410
  Tx Bytes:   3441480
  TxOffered:  3.05909 Mbps
  Rx Packets: 1852
  Rx Bytes:   2644656
  Throughput: 2.35081 Mbps
Flow 2 (19.71.8.3 -> 19.71.8.4)
  Tx Packets: 2410
  Tx Bytes:   3441480
  TxOffered:  3.05909 Mbps
  Rx Packets: 1894
  Rx Bytes:   2704632
  Throughput: 2.40412 Mbps
amirhosein@qemu:~/ns3/ns-allinone-3.30/ns-3.30$
```

CTS/RTS doesn't play a role for this part because there isn't any collision, and by enabling CTS/RTS you're just adding an unwanted delay, as you can see from the result it decreased the throughput and there is no need to enable it.

Screenshot of the topology in Netanim : (also RTS/CTS packet is visible)



b) Pcap files are located in 802.11\ Simulation/b/Wifi_PCAP_i.pcap, and one of them is like the screenshot down here :



As it can be seen there are four types of protocols,

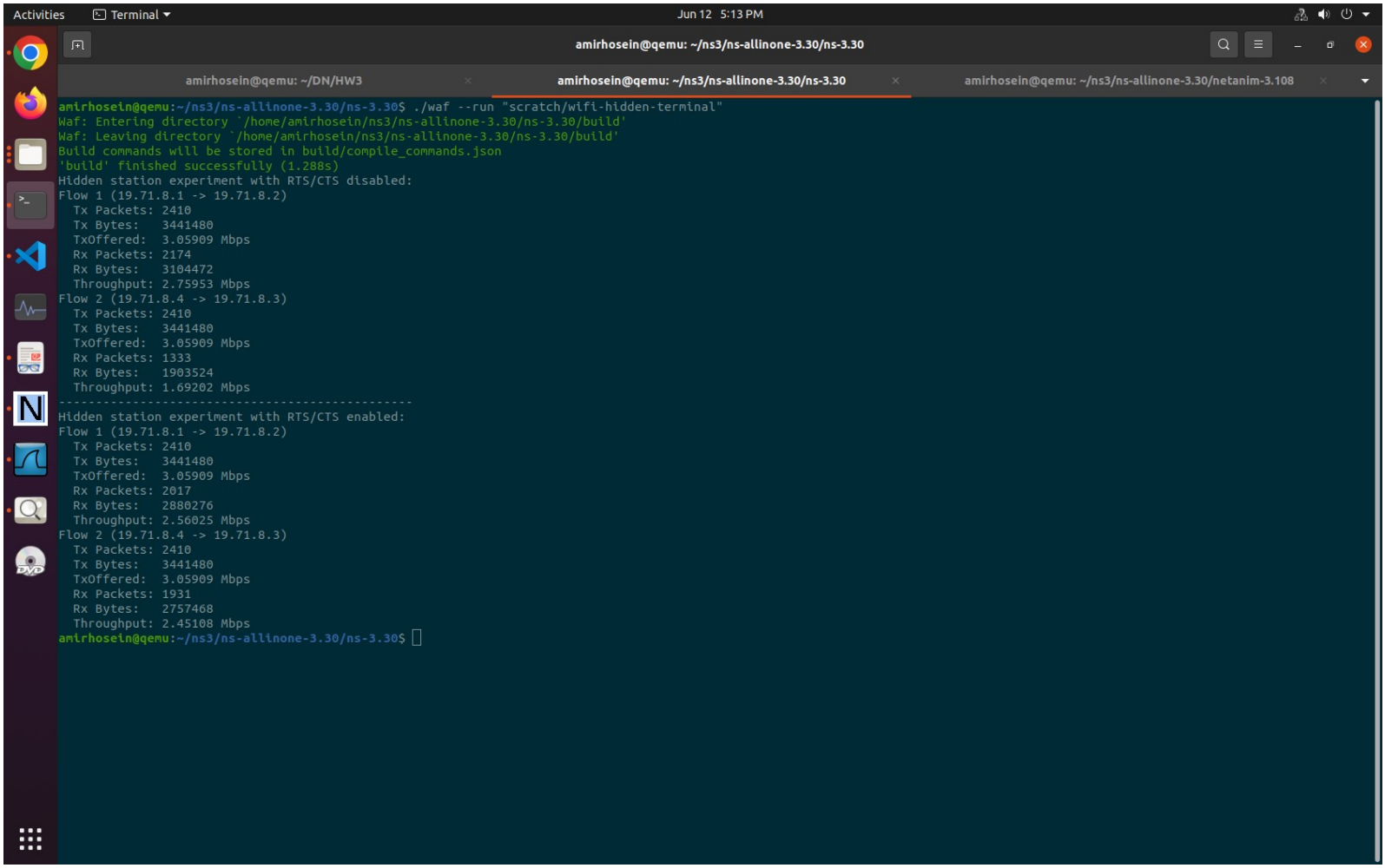
1) ARP: which is used for getting the mac addresses,

2) 802.11: which is the acknowledge packet to the ICMP protocol,

3) ICMP: the main request for pinging the destination,

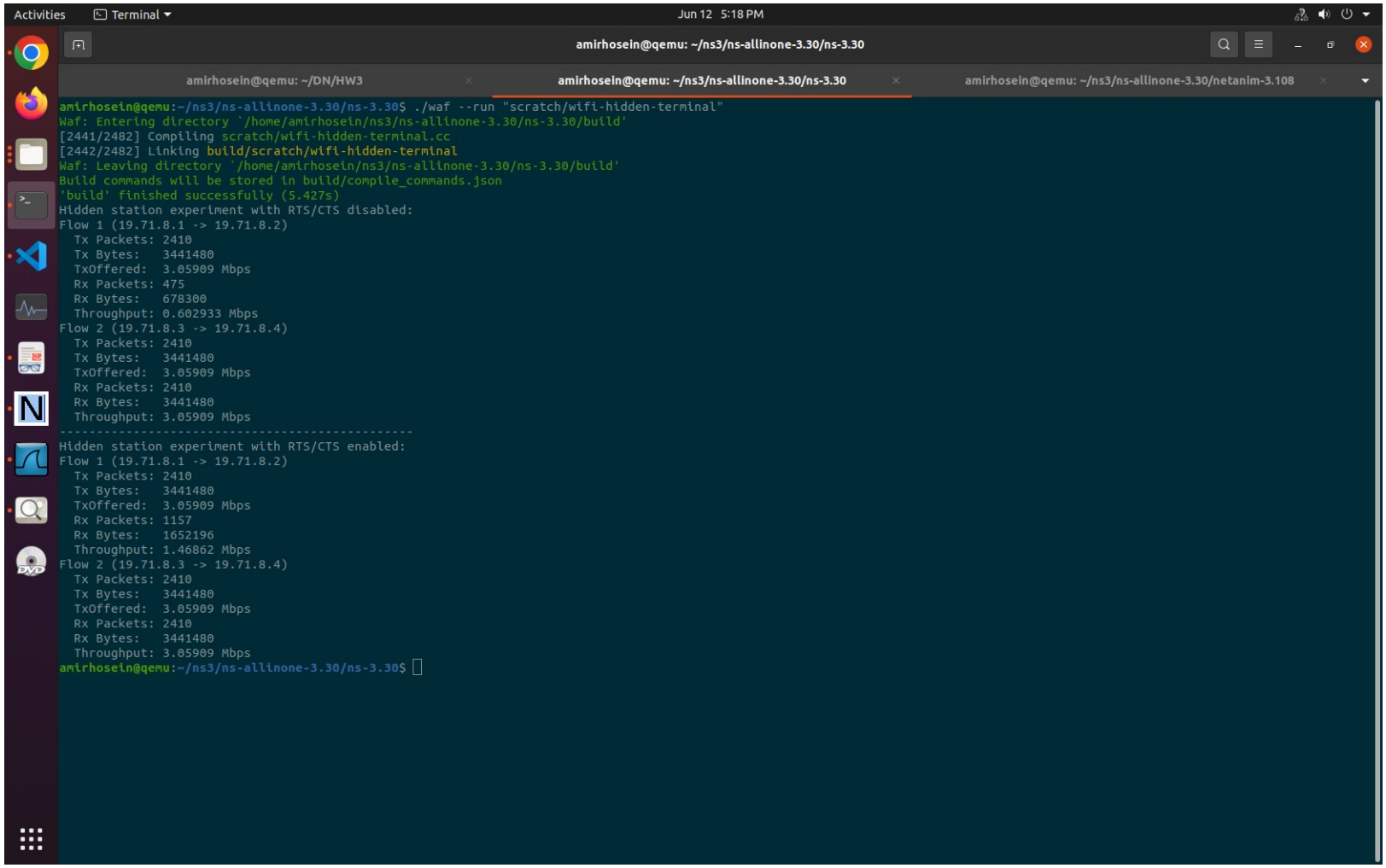
4) UDP: packets from Bi to Ai nodes.

c) As it can be seen, results are improved a bit because receivers might face collision due to the ack of one side and the packet of the other one and with RTS/CTS enabled overall throughput would definitely increase.

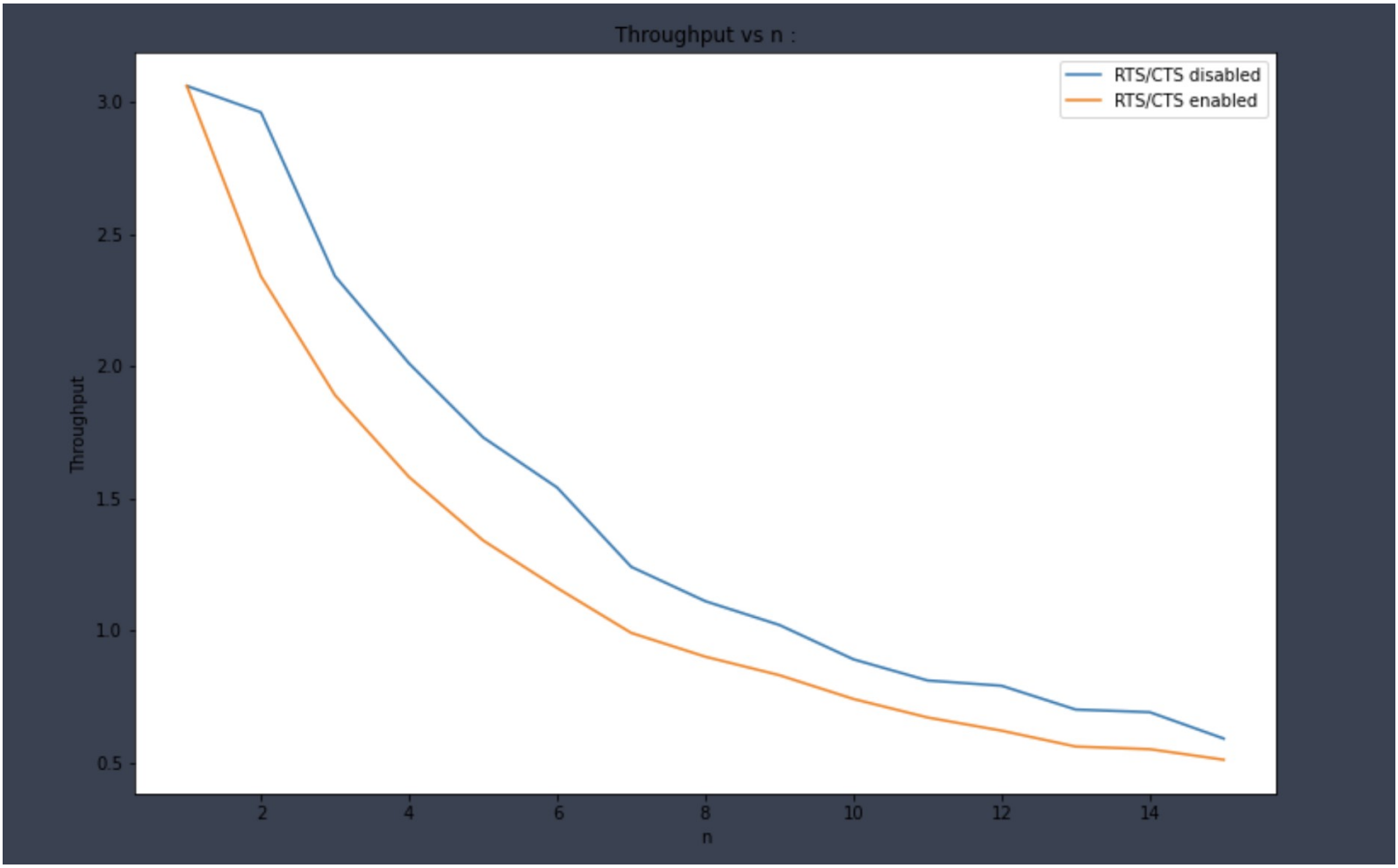


d) When only one flow changes there is going to be a really though situation for B1 because when it wants to get the packet from A1, B2 is also sending another packet simultaneously and it's going to make collision all the time, but with RTS/CTS enabled it's not going to happen because node B2 is listening and waits for free time to send the packet and the final throughput of flow one is increased by the factor of two!

So it is obviously a good choice to use RTS/CTS here.



e) Screenshots are located in 802\ Simulation/e/Wifi_Result_full_i.png, and for getting more of the terminal output I changed the resolution to 4K and some of the results are in two images, in the end the average throughput is plotted over n and as you can see throughput is decreased by increasing n, and the reason for such behavior is that by increasing n links between each node B is also increased exponentially and that means probability of collision is increased as well, so overall throughput must be lowered.



Also with RTS/CTS enabled you should expect lower throughput, because there are some added packets for checking the situation of channels and that results to lower throughput in the end.

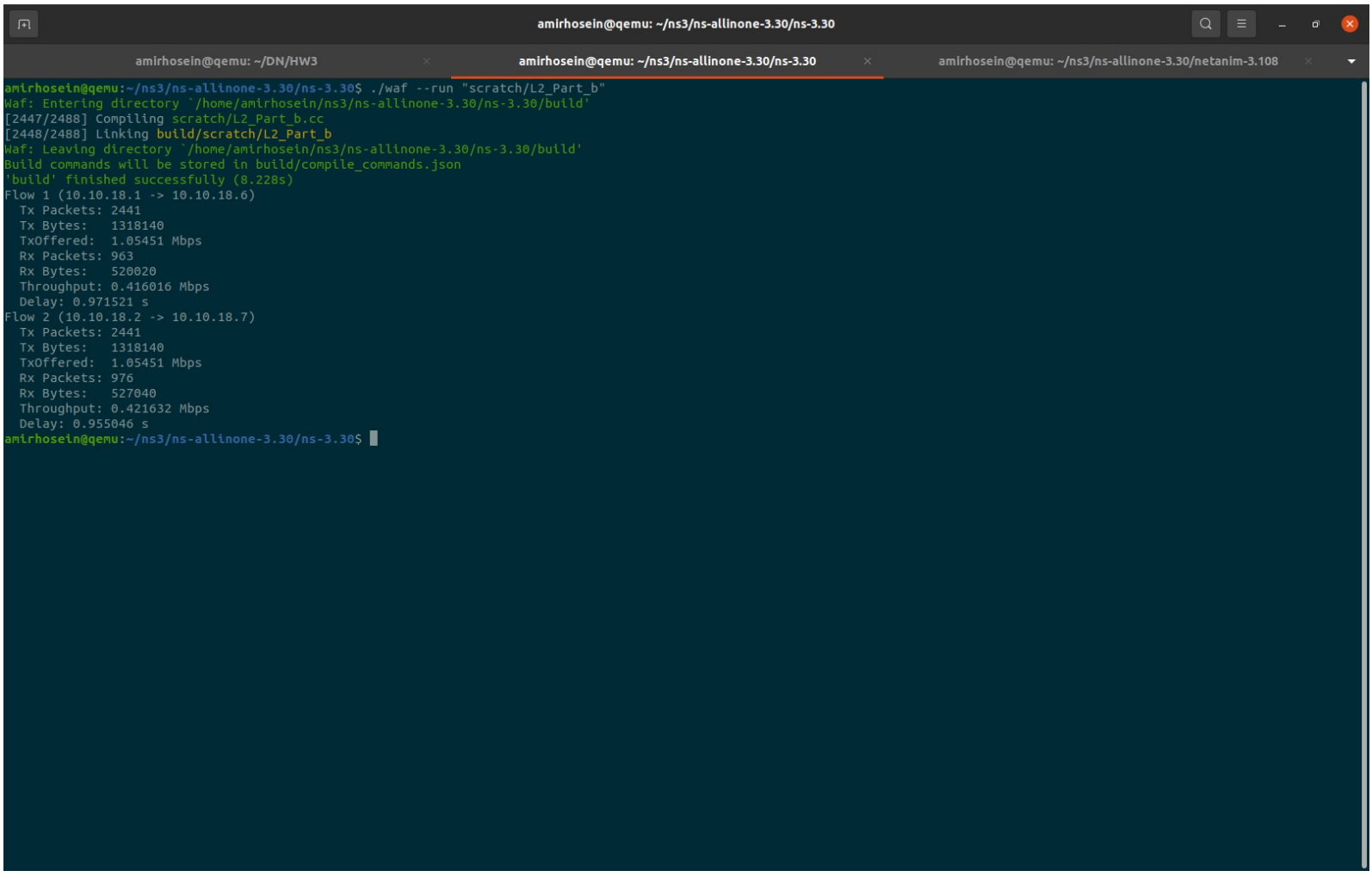
Simple Layer 2 System :

a) For implementation of this part I used the link below :

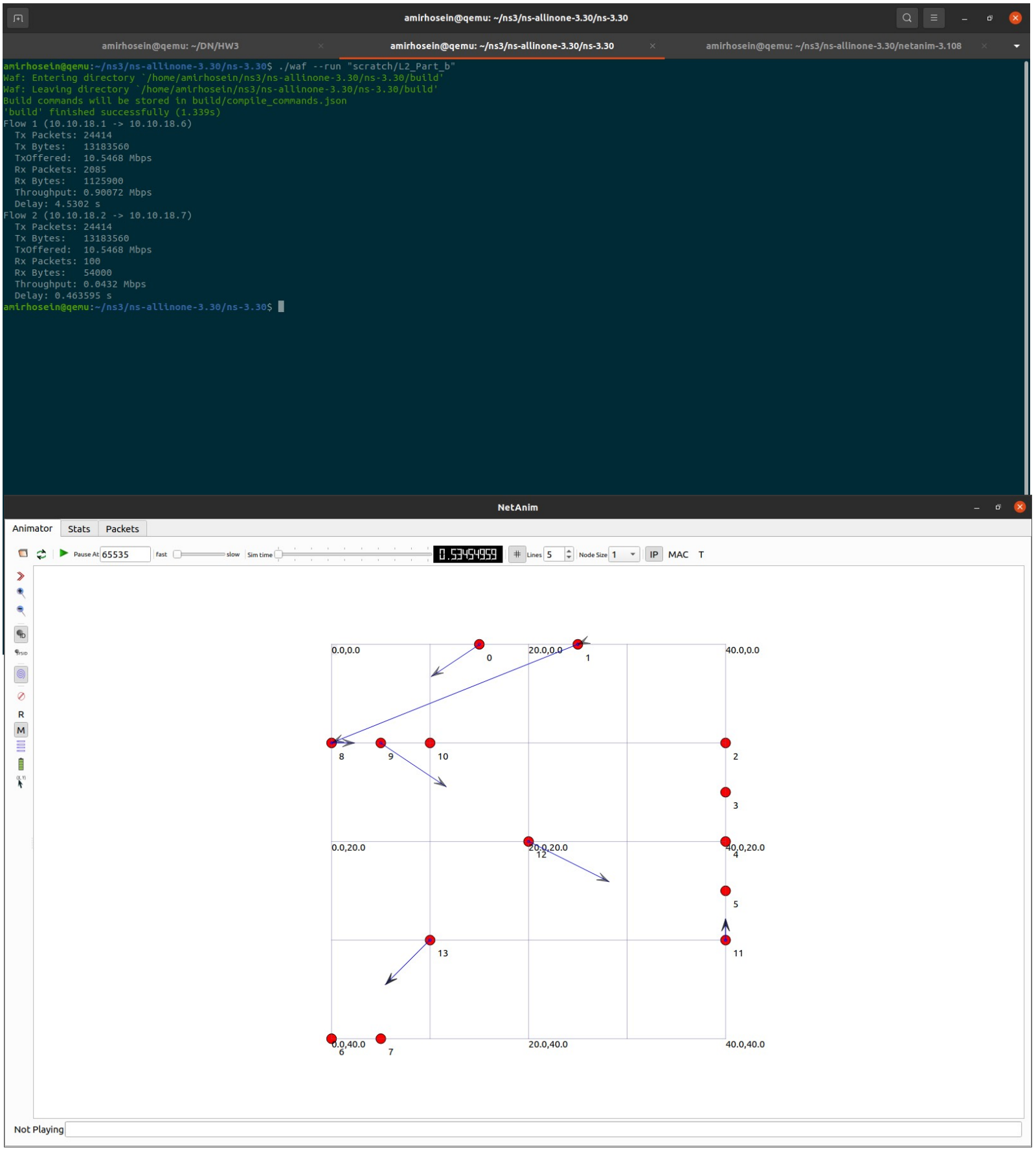
https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-using-stl-in-c/

it simply gets the graph of your network and returns a vector of links in Minimum Spanning Tree (MST) using kruskal's algorithm, and I used the output vector in part b to create the topology.

b) for this part I link computers and then iterate over links in MST to create links between bridges, in the end two onoff flows created and the result is shown below :



c) when you change packet generation rate to 10000kbps you face a ton of collision and also because flows are using a shared link with maximum capacity of 1Mbps you end up having a really low throughput.(also due to symmetry they should be equal but it seems that it’s a bug that ns3 doesn’t work well without/with mobility and for others they happened to be equal after adding mobility but for me it didn’t!)



d) (Bonus) For this part I Implemented Dijkstra algorithm with the root node of c1 instead of kruskal's MST algorithm and it generated a new path without shared links, so every flow could use all the resources of the physical channel which is 1Mbps, no matter what is the generation rate of packets are.