

AQM-LLM

This document is intended for IoT lab researchers. For details, please refer to the AQM-LLM documentation.

This document is used to quickly build the experimental structure, not to quickly complete all experiments.

Preparation

There are 3 VMs in total:

VM_BBR2: Contains BBR2 congestion control algorithm

VM_BBR3: Contains BB3 congestion control algorithm

VM_L4S: Contains everything about the L4S architecture, but only for the VMs for Client and Server (**TCP Prague congestion control algorithm and ECN enabled**). Routers need to manually install the L4S architecture and enable **DualPI2 AQM**.

The VM backup is only accessible to the Deakin account, and I don't have permission to change it to anyone else.

VM_BBR2 OneDrive link: [VM bbr2](#)

VM_BBR3 OneDrive link: [VM bbr3](#)

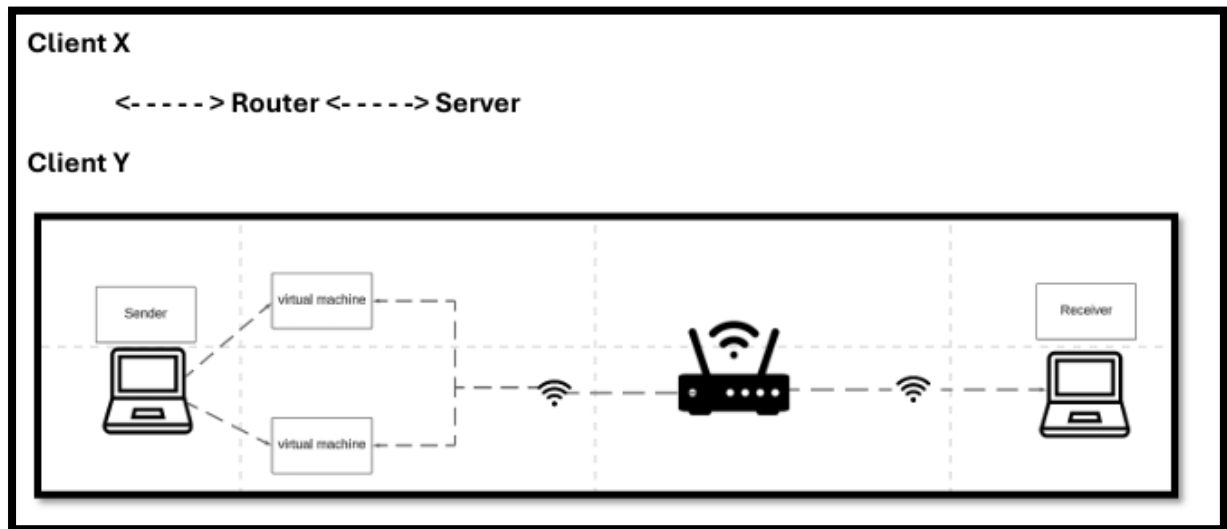
VM_L4S OneDrive link: [Ubuntu 64 prague](#)

The su password and login password for the VM are both **SIT7232024**

Process

Hardware: 2 Windows computers (PC1 and PC2), 1 Ubuntu computer or Linux-supported router.

Software: VMware workstation Pro (17th generation is a personal free version, I use 16th generation, any differences need to be solved by yourself.)



Roles: Client, Router, Server.

Client X is a client that supports L4S architecture, and Client Y is a client that does not support L4S architecture.

Client X uses VM_L4S, and Client Y uses VM_BBR, VM_BBR2, or VM_BBR3 according to your experimental requirements.

Router requires you to install L4S architecture on MikroTik router or Ubuntu system computer.

Server uses VM_L4S.

Open VMware workstation Pro on PC1 and PC2 and start VM_L4S on both VMware.

Enter the following command in the VMs terminal

Server side:

iperf3 -s -p 3000

iperf3 -s -p 3001

Client X side:

iperf3 -c [ip address] -p 3000 -C prague -tinf

Client Y side:

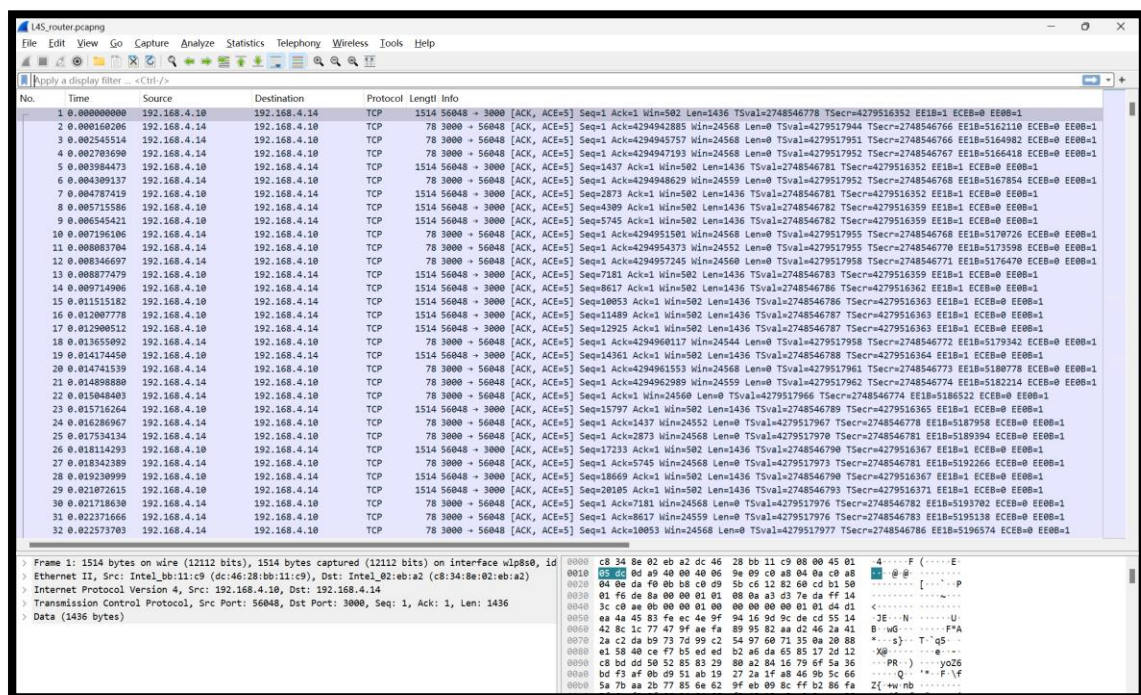
iperf3 -c [ip address] -p 3001 -C cubic -tinf

[ip address] is the IP address of your Server side.

Launch Wireshark on Router or Server side, to check L4S is enable or not.

After iperf3 connects properly, please keep Clients connected to Server. Then start Wireshark on Server side or Router side, after opening Wireshark enter

ip.dsfield.ecn == 3



As long as you can see the result after typing it, it means that ECN=3 marked packets appeared, it also means that DualPI2 is marking the congested packets with ECN. please let iperf3 connect for a longer period of time, in my previous tests congestion appeared at a ratio of about 1:6000 or 20,000th packets before the first congestion marking appeared.

No.	Time	Source	Destination	Protocol	Length	Info
1439..	111.243267878	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62234593 Win=24551 Len=0 TSval=4279629176 TSecr=2748656102 EEB=312258 ECEB=0 EEOB=1
1439..	111.251783295	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=622346029 Win=24542 Len=0 TSval=4279629176 TSecr=2748656102 EEB=313686 ECEB=0 EEOB=1
1439..	111.252620772	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62237465 Win=24534 Len=0 TSval=4279629176 TSecr=2748656104 EEB=315122 ECEB=0 EEOB=1
1439..	111.253463333	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62238901 Win=24525 Len=0 TSval=4279629176 TSecr=2748656105 EEB=316558 ECEB=0 EEOB=1
1439..	111.254070487	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62240337 Win=24517 Len=0 TSval=4279629176 TSecr=2748656106 EEB=317994 ECEB=0 EEOB=1
1439..	111.254425864	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62241773 Win=24508 Len=0 TSval=4279629176 TSecr=2748656107 EEB=319430 ECEB=0 EEOB=1
1439..	111.254851513	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62244081 Win=24551 Len=0 TSval=4279629180 TSecr=2748656111 EEB=323738 ECEB=0 EEOB=1
1439..	111.264981311	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62247517 Win=24542 Len=0 TSval=4279629180 TSecr=2748656112 EEB=325174 ECEB=0 EEOB=1
1439..	111.265841537	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=62248953 Win=24534 Len=0 TSval=4279629180 TSecr=2748656113 EEB=326610 ECEB=0 EEOB=1
2425..	222.559662377	192.168.4.10	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=97591133 Win=24551 Len=0 TSval=4279740506 TSecr=2748770665 EEB=2114358 ECEB=0 EEOB=1
4714..	480.751815747	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=191472505 Win=24526 Len=0 TSval=4279918700 TSecr=2748948559 EEB=12109658 ECEB=0 EEOB=1
4714..	480.752755491	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=191473941 Win=24515 Len=0 TSval=4279918700 TSecr=2748948562 EEB=12111086 ECEB=0 EEOB=1
4813..	480.432754123	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=195649029 Win=24551 Len=0 TSval=4279926382 TSecr=2748956343 EEB=16286974 ECEB=0 EEOB=1
4813..	480.432942194	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=195651265 Win=24542 Len=0 TSval=4279926382 TSecr=2748956343 EEB=16288410 ECEB=0 EEOB=1
5964..	497.892052799	192.168.4.10	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=242328445 Win=24551 Len=0 TSval=4280015634 TSecr=2749045869 EEB=12633942 ECEB=0 EEOB=1
5964..	497.892599006	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=242329881 Win=24542 Len=0 TSval=4280015634 TSecr=2749045871 EEB=12635378 ECEB=0 EEOB=1
5968..	498.190218473	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=242508507 Win=24551 Len=0 TSval=4280016138 TSecr=2749046358 EEB=12818570 ECEB=0 EEOB=1
5968..	498.190368758	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=242508509 Win=24542 Len=0 TSval=4280016138 TSecr=2749046360 EEB=12812006 ECEB=0 EEOB=1
6023..	565.430854656	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278404053 Win=24524 Len=0 TSval=4280083373 TSecr=2749113270 EEB=15235110 ECEB=0 EEOB=1
6023..	565.430858593	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278404053 Win=24513 Len=0 TSval=4280083373 TSecr=2749113272 EEB=15236554 ECEB=0 EEOB=1
6023..	565.430824723	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278486925 Win=24502 Len=0 TSval=4280083373 TSecr=2749113275 EEB=15237990 ECEB=0 EEOB=1
6023..	565.430436927	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278488361 Win=24491 Len=0 TSval=4280083373 TSecr=2749113275 EEB=15239426 ECEB=0 EEOB=1
6023..	565.433648967	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278492669 Win=24551 Len=0 TSval=4280083373 TSecr=2749113287 EEB=15243734 ECEB=0 EEOB=1
6023..	565.434608135	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278494105 Win=24542 Len=0 TSval=4280083373 TSecr=2749113289 EEB=15245170 ECEB=0 EEOB=1
6023..	565.430443383	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278495541 Win=24534 Len=0 TSval=4280083373 TSecr=2749113291 EEB=15246606 ECEB=0 EEOB=1
6023..	565.435316795	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278496977 Win=24525 Len=0 TSval=4280083373 TSecr=2749113294 EEB=15248042 ECEB=0 EEOB=1
6023..	565.435997253	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278498413 Win=24517 Len=0 TSval=4280083373 TSecr=2749113308 EEB=15249478 ECEB=0 EEOB=1
6023..	565.437608708	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=278499849 Win=24508 Len=0 TSval=4280083373 TSecr=2749113328 EEB=15250914 ECEB=0 EEOB=1
7378..	608.166721618	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=301737701 Win=24551 Len=0 TSval=4280126101 TSecr=2749155912 EEB=4932398 ECEB=0 EEOB=1
7378..	608.166851692	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=301737701 Win=24542 Len=0 TSval=4280126101 TSecr=2749155915 EEB=4933834 ECEB=0 EEOB=1
7642..	630.401802673	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=312869973 Win=24551 Len=0 TSval=4280148335 TSecr=2749178018 EEB=16065706 ECEB=0 EEOB=1
7642..	630.401818173	192.168.4.14	192.168.4.10	TCP	78	3000 → 56048 [ACK, ACE=5] Seq=1 Ack=312870549 Win=24542 Len=0 TSval=4280148335 TSecr=2749178021 EEB=16067142 ECEB=0 EEOB=1

```

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens3, id 0
> Ethernet II, Src: VMware_2e:29:97 (08:0c:29:2e:29:97), Dst: Intel_02:eb:a2 (c8:34:8e:02:eb:a2)
> Internet Protocol Version 4, Src: 192.168.4.10, Dst: 192.168.4.14
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x01 (DSCP: CS0, ECN: ECT(1))
    0000 00.. = Differentiated Services Codepoint: Default (0)
      .... 01 = Explicit Congestion Notification: ECN-Capable Transport codepoint '01' (1)
  Total Length: 60
  Identification: 0xd904 (55556)
  010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: TCP (6)
  Header Checksum: 0xd84d [validation disabled]
  0000 c8 34 8e 02 eb a2 00 c 29 2e 29 97 08 00 45 01 4.....).---E
  0010 00 3c d9 04 40 00 00 06 d8 4d c0 a8 04 0a c0 a8 <---@-M-----
  0020 04 0e e6 aa 0b b8 7f 9b c6 3b 00 00 00 00 a1 c2 ;-----;
  0030 fa f0 89 97 00 00 02 04 05 b4 04 02 08 0a a2 2c ,-----,
  0040 ac 3d 00 00 00 00 01 03 03 07 =-----
  
```

Checkpoint: If you can see packets with ECN=3, it means that the L4S architecture has been enabled. You have completed the setup part of the experiment. The next step is to collect L4S data, convert the collected data into PKL experience pool files that LLM can understand, and finally train in RunPod.

Next, please follow the AQM-LLM Documentation to download the files required for the LLM model and NetLLM architecture. (If you are in a hurry, please seek help from Deol in the IoT lab. The workstation computer theoretically has the complete NetLLM + LLM files.)

After you download the Llama2-7b-hf and NetLLM files, the structure of the entire trained model should look like this:



Open the NetLLM folder:

	adaptive_bitrate_streaming	2024/9/10 13:50	文件夹	
	downloaded_plms	2024/9/10 13:50	文件夹	
	.gitignore	2024/7/27 15:31	Git Ignore 源文件	1 KB
	LICENSE	2024/7/27 15:31	文件	2 KB
	README.md	2024/7/27 15:31	Markdown 源文件	1 KB

Adaptive_bitrate_steaming is all the files needed to train LLM and downloaded_plms is the location where the LLM (Llama2-7b-hf) model is stored. The directory where the LLM is saved is:



NetLLM > downloaded_plms > llama > base >				
排序 查看 ...				
名称	修改日期	类型	大小	
	2024/9/10 14:14	文件夹		
	2024/8/13 19:49	Git Attributes 源...	2 KB	
	2024/8/13 19:49	JSON 源文件	1 KB	
	2024/8/13 19:49	JSON 源文件	1 KB	
	2024/8/13 19:49	文本文档	7 KB	
	2024/8/13 19:49	JSON 源文件	27 KB	
	2024/8/13 20:03	SAFETENSORS ...	9,742,753...	
	2024/8/13 19:55	SAFETENSORS ...	3,418,260...	
	2024/8/13 19:49	JSON 源文件	27 KB	
	2024/8/13 20:04	BIN 文件	9,742,808...	
	2024/8/13 19:58	BIN 文件	3,418,277...	
	2024/8/13 19:49	Markdown 源文件	22 KB	
	2024/8/13 19:49	Microsoft Edge ...	1,224 KB	
	2024/8/13 19:49	JSON 源文件	1 KB	
	2024/8/13 19:49	JSON 源文件	1,800 KB	
	2024/8/13 10:40	MODEL 文件	480 KB	

The process of renting RunPod is not explained here. If you have a local GPU that can train NetLLM, you don't need to rent RunPod. It is recommended to read the process of AQM-LLM Documentation before quick operation.

Use the scp command to upload the NetLLM folder to RunPod. Be sure to upload it to cd /workspace, otherwise all storage contents will be erased when you close RunPod. cd to the adaptive_bitrate_streaming folder and use the following command:

```
python -m pip install --upgrade pip && pip install openprompt==1.0.1 && pip install numpy==1.24.4 && pip install peft==0.6.2 && pip install transformers==4.34.1 && pip install --upgrade huggingface_hub && pip install scikit-learn && pip install munch
```

```
python run_plm.py --adapt --grad-accum-steps 32 --plm-type llama --plm-size base --rank 128 --device cuda:0 --lr 0.0001 --warmup-steps 2000 --num-epochs 80 --eval-per-epoch 2
```

This process is trained entirely using the official NetLLM file, just to familiarize you with how the entire NetLLM and LLM work. You don't have to run 80 iterations.

Next, you need to modify the PKL file yourself, modify other codes to remove the restrictions on the model's PKL content, and then train. The specific process AQM-LLM Documentation has explained it.