Title

Performance Analysis of LTE by setting up a Testbed using srsLTE

Students:
Mahanth Kumar Valluri CS22MTECH11015
Vishesh Kothari CS22MTECH12004
Tejas Deshmukh CS22MTECH12005

Problem statement:

Study of LTE using srsLTE, by setting up a test bed

Project description :

First we installed srsLTE and using the hardware USRP B210 and antennas we set up the LTE testbed .We analyzed the throughput of the channel using “iperf” and we have used ping to flood ICMP packets by keeping the overall data sent as constant.Then we kept the time for which the flood ping was running as constant for different packet sizes and observed a change in SNR values.
Then we introduced the concept of NUMA to analyze throughput by explicitly allocating cores to EPC and eNB on either the same or different nodes. Stress-ng is used to increase the load on remaining non-allocated cores and throughput was analyzed for this scenario.

Implementation

Hardware requirements:

- USRP B210(1 for eNB,ePC and 1 for UE)
- Two Monitors
- Antennas-**VERT2450** (for USRP B210)

Software Requirements:

- srsLTE

Tools

Iperf3:

iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. It works by generating traffic from a computer acting as a client which is sent to the IP address of a computer acting as the server
Syntax:

```
iperf3 -c <ip_address>
Iperf3 -s
```

Ping:

It is used to check whether the destination ip-address exists and is able to accept requests.
syntax:

```
ping [options] hostname
```

Taskset:
Using the "**taskset**" command tool, the user can fetch or set the CPU affinity of a particular process with its given process id (PID) and also CPU cores can be assigned manually.

Syntax:
Taskset [options] mask command [argument..]

Link : https://linuxhint.com/use-taskset-command/

Pqos:
The pqos tool provides support to set up the Intel CAT (Cache Allocation Technology) capabilities, and monitor last level cache occupancy via CMT (Cache Monitoring Technology) and monitor memory bandwidth via MBM (Memory Bandwidth Monitoring).
https://manpages.ubuntu.com/manpages/xenial/man8/pqos.8.html#:~:text=The%20pqos%20tool%20provides%20support,MBM%20(Memory%20Bandwidth%20Monitoring).

Stress-ng
Stress-ng is used to increase load on the CPU cores
https://manpages.ubuntu.com/manpages/bionic/man1/stress-ng.1.html

Numactl: controls NUMA policy on processes and shared memory
numactl –hardware : show inventory of available nodes on the system

Installation of srsLTE
Firstly the required libraries are installed using the following command.

```
sudo     apt-get     install     build-essential     cmake     libfftw3-dev     libmbedtls-dev
libboost-program-options-dev libconfig++-dev libsctp-dev
```

Then we got the official repository of srsRAN(srsLTE) and changed the head to branch release 20.0.1. Then build the srsLTE. At the time of "make" command we added j8 at the end which means that we are assigning 8 processors of the total 12 processors in order to perform the given task.

```
git clone https://github.com/srsRAN/srsRAN.git
git checkout release-20.0.1
```

```
cd srsRAN
mkdir build
cd build
cmake ../
make -j8
make test
```

This installs srsRAN and also copies the default srsRAN config files to ~/.config/srsran.

```
./srslte_install_configs.sh user
```

These two commands are used to locate and configure the attached peripherals(USRP B210).

```
sudo uhd_find_devices
```

Images will be downloaded, that is there are image packages for a desired device and configuration.Also users have the option to specify which images to download

```
sudo uhd_images_downloader
```

The following command is to start epc on a host

```
sudo srsepc
```

The following command is used to start eNB on a host using one USRP B210

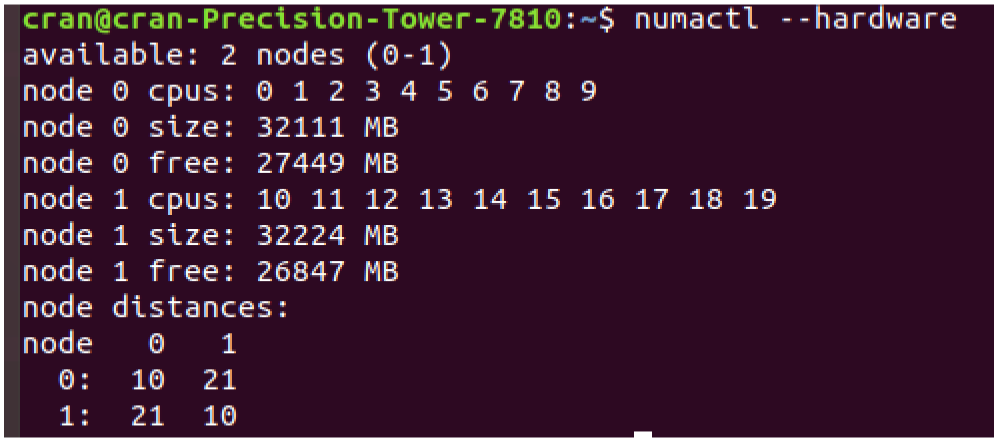
```
sudo srsenb
```

The following command is used to start User on a host using one USRP B210

```
sudo srsue
```

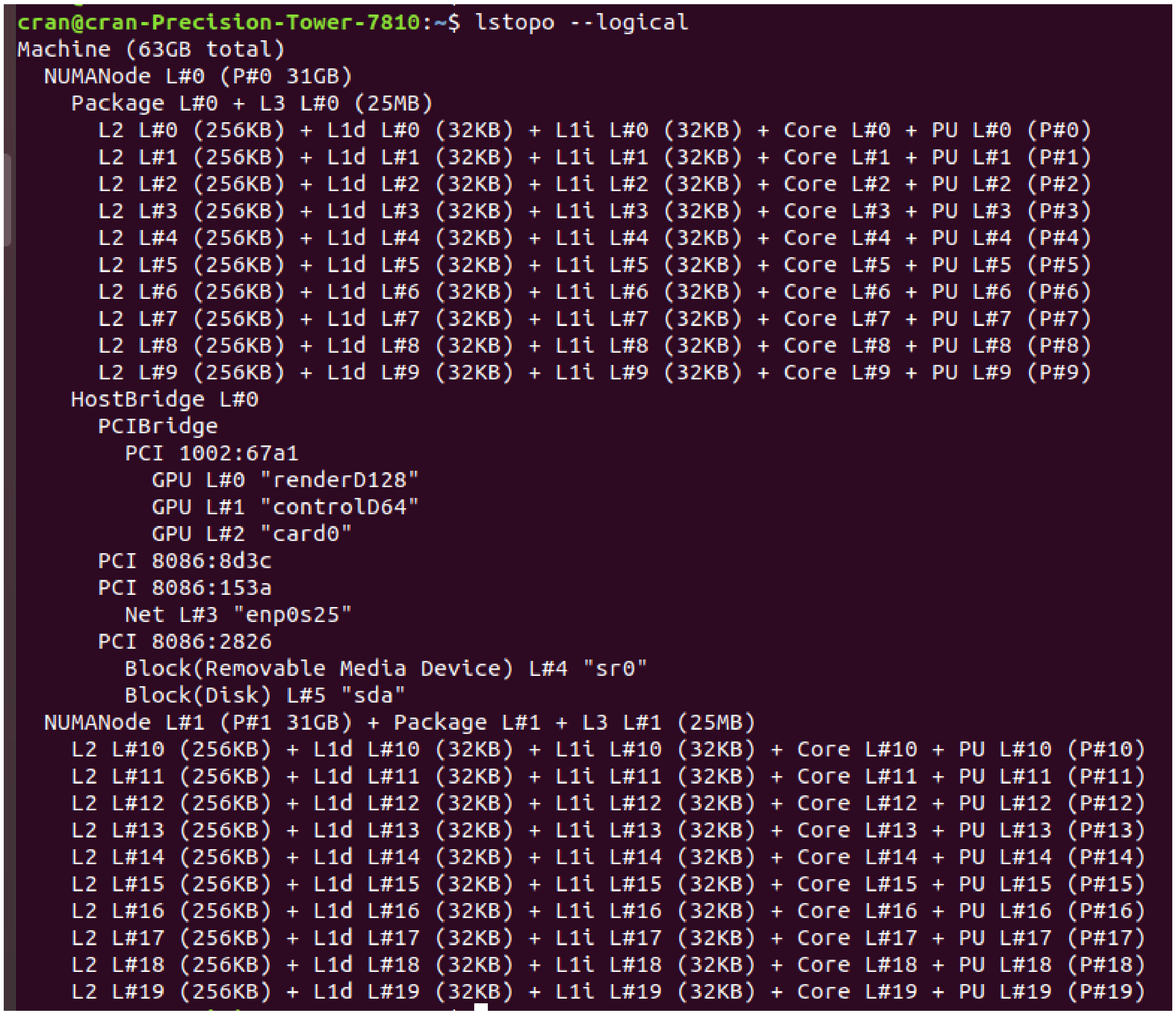
<u>The setup of srsLTE is completed with the above commands.</u>

To check for **NUMA**, on a system:



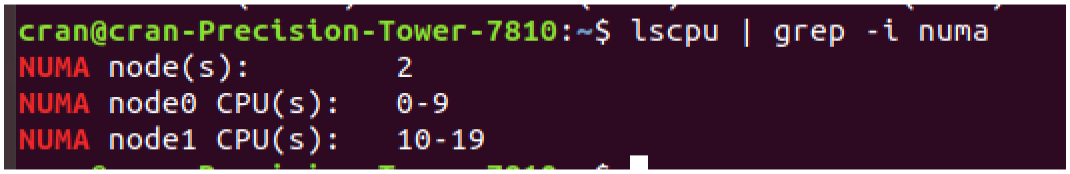
```
cran@cran-Precision-Tower-7810:~$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9
node 0 size: 32111 MB
node 0 free: 27449 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19
node 1 size: 32224 MB
node 1 free: 26847 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

To understand the **NUMA topology** of cpu's:



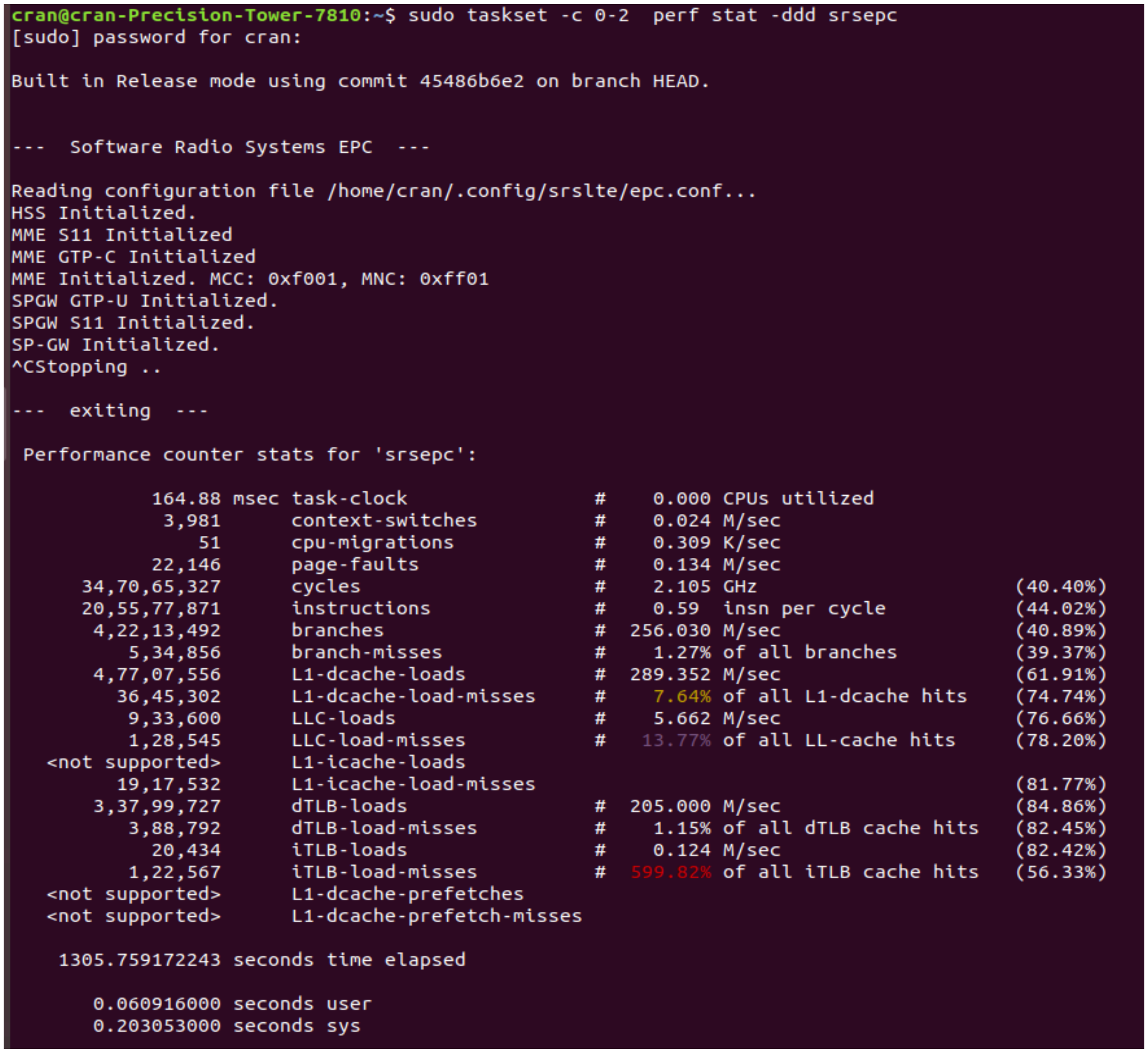
```
cran@cran-Precision-Tower-7810:~$ lstopo --logical
Machine (63GB total)
  NUMANode L#0 (P#0 31GB)
    Package L#0 + L3 L#0 (25MB)
      L2 L#0 (256KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)
      L2 L#1 (256KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1 + PU L#1 (P#1)
      L2 L#2 (256KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2 + PU L#2 (P#2)
      L2 L#3 (256KB) + L1d L#3 (32KB) + L1i L#3 (32KB) + Core L#3 + PU L#3 (P#3)
      L2 L#4 (256KB) + L1d L#4 (32KB) + L1i L#4 (32KB) + Core L#4 + PU L#4 (P#4)
      L2 L#5 (256KB) + L1d L#5 (32KB) + L1i L#5 (32KB) + Core L#5 + PU L#5 (P#5)
      L2 L#6 (256KB) + L1d L#6 (32KB) + L1i L#6 (32KB) + Core L#6 + PU L#6 (P#6)
      L2 L#7 (256KB) + L1d L#7 (32KB) + L1i L#7 (32KB) + Core L#7 + PU L#7 (P#7)
      L2 L#8 (256KB) + L1d L#8 (32KB) + L1i L#8 (32KB) + Core L#8 + PU L#8 (P#8)
      L2 L#9 (256KB) + L1d L#9 (32KB) + L1i L#9 (32KB) + Core L#9 + PU L#9 (P#9)
    HostBridge L#0
      PCIBridge
        PCI 1002:67a1
          GPU L#0 "renderD128"
          GPU L#1 "controlD64"
          GPU L#2 "card0"
      PCI 8086:8d3c
      PCI 8086:153a
        Net L#3 "enp0s25"
      PCI 8086:2826
        Block(Removable Media Device) L#4 "sr0"
        Block(Disk) L#5 "sda"
  NUMANode L#1 (P#1 31GB) + Package L#1 + L3 L#1 (25MB)
    L2 L#10 (256KB) + L1d L#10 (32KB) + L1i L#10 (32KB) + Core L#10 + PU L#10 (P#10)
    L2 L#11 (256KB) + L1d L#11 (32KB) + L1i L#11 (32KB) + Core L#11 + PU L#11 (P#11)
    L2 L#12 (256KB) + L1d L#12 (32KB) + L1i L#12 (32KB) + Core L#12 + PU L#12 (P#12)
    L2 L#13 (256KB) + L1d L#13 (32KB) + L1i L#13 (32KB) + Core L#13 + PU L#13 (P#13)
    L2 L#14 (256KB) + L1d L#14 (32KB) + L1i L#14 (32KB) + Core L#14 + PU L#14 (P#14)
    L2 L#15 (256KB) + L1d L#15 (32KB) + L1i L#15 (32KB) + Core L#15 + PU L#15 (P#15)
    L2 L#16 (256KB) + L1d L#16 (32KB) + L1i L#16 (32KB) + Core L#16 + PU L#16 (P#16)
    L2 L#17 (256KB) + L1d L#17 (32KB) + L1i L#17 (32KB) + Core L#17 + PU L#17 (P#17)
    L2 L#18 (256KB) + L1d L#18 (32KB) + L1i L#18 (32KB) + Core L#18 + PU L#18 (P#18)
    L2 L#19 (256KB) + L1d L#19 (32KB) + L1i L#19 (32KB) + Core L#19 + PU L#19 (P#19)
```

To get cores that are present on each node



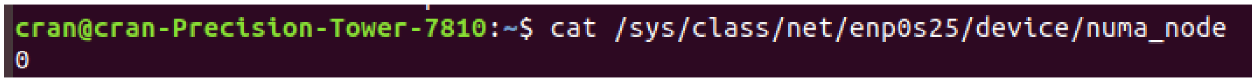
```
cran@cran-Precision-Tower-7810:~$ lscpu | grep -i numa
NUMA node(s):          2
NUMA node0 CPU(s):     0-9
NUMA node1 CPU(s):     10-19
```

An example of the taskset command is given in the screenshot below:
c = cores given
perf stat = performance stats
ddd : in detail;
so after you stop this instance, we will get performance stats for that instance(eNB/ePC)



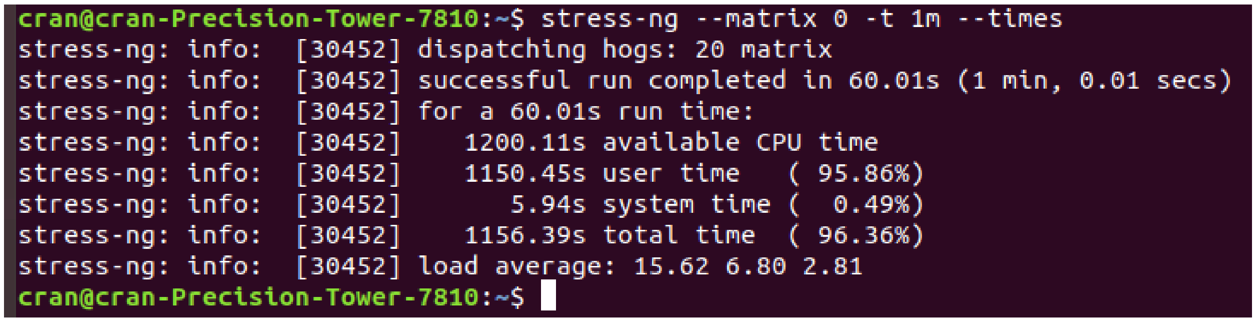
```
cran@cran-Precision-Tower-7810:~$ sudo taskset -c 0-2  perf stat -ddd srsepc
[sudo] password for cran:

Built in Release mode using commit 45486b6e2 on branch HEAD.


---  Software Radio Systems EPC  ---

Reading configuration file /home/cran/.config/srslte/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf001, MNC: 0xff01
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
^CStopping ..

---  exiting  ---

 Performance counter stats for 'srsepc':

            164.88 msec task-clock                #    0.000 CPUs utilized
             3,981      context-switches          #    0.024 M/sec
                51      cpu-migrations            #    0.309 K/sec
            22,146      page-faults               #    0.134 M/sec
      34,70,65,327      cycles                    #    2.105 GHz                      (40.40%)
      20,55,77,871      instructions              #    0.59  insn per cycle           (44.02%)
       4,22,13,492      branches                  #  256.030 M/sec                    (40.89%)
         5,34,856       branch-misses             #    1.27% of all branches          (39.37%)
       4,77,07,556      L1-dcache-loads           #  289.352 M/sec                    (61.91%)
         36,45,302      L1-dcache-load-misses     #    7.64% of all L1-dcache hits    (74.74%)
          9,33,600      LLC-loads                 #    5.662 M/sec                    (76.66%)
          1,28,545      LLC-load-misses           #   13.77% of all LL-cache hits     (78.20%)
   <not supported>      L1-icache-loads
         19,17,532      L1-icache-load-misses                                         (81.77%)
       3,37,99,727      dTLB-loads                #  205.000 M/sec                    (84.86%)
          3,88,792      dTLB-load-misses          #    1.15% of all dTLB cache hits   (82.45%)
            20,434      iTLB-loads                #    0.124 M/sec                    (82.42%)
          1,22,567      iTLB-load-misses          #  599.82% of all iTLB cache hits   (56.33%)
   <not supported>      L1-dcache-prefetches
   <not supported>      L1-dcache-prefetch-misses

    1305.759172243 seconds time elapsed

       0.060916000 seconds user
       0.203053000 seconds sys
```

To determine if the NIC is connected to which NUMA node and below we can see that it is connected to node 0



```
cran@cran-Precision-Tower-7810:~$ cat /sys/class/net/enp0s25/device/numa_node
0
```

Introducing noisy neighbor



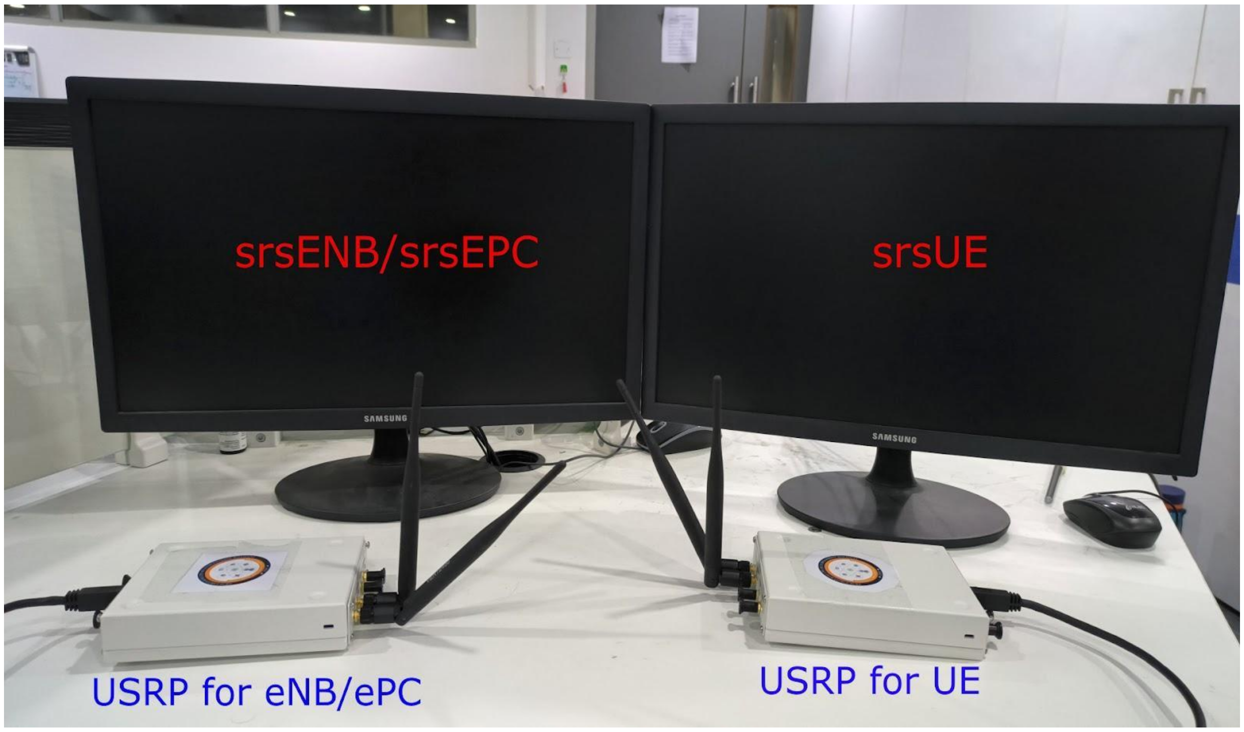
```
cran@cran-Precision-Tower-7810:~$ stress-ng --matrix 0 -t 1m --times
stress-ng: info:  [30452] dispatching hogs: 20 matrix
stress-ng: info:  [30452] successful run completed in 60.01s (1 min, 0.01 secs)
stress-ng: info:  [30452] for a 60.01s run time:
stress-ng: info:  [30452]    1200.11s available CPU time
stress-ng: info:  [30452]    1150.45s user time   ( 95.86%)
stress-ng: info:  [30452]       5.94s system time (  0.49%)
stress-ng: info:  [30452]    1156.39s total time  ( 96.36%)
stress-ng: info:  [30452] load average: 15.62 6.80 2.81
cran@cran-Precision-Tower-7810:~$
```

Challenges Faced:

- Fluctuations in signal leading to not so accurate results
- Real time analysis of observations and comparison with theoretical observations.
- Drop in SNR while increasing the packet size during flood ping request

Experimental Setup

Test Scenario 1 :

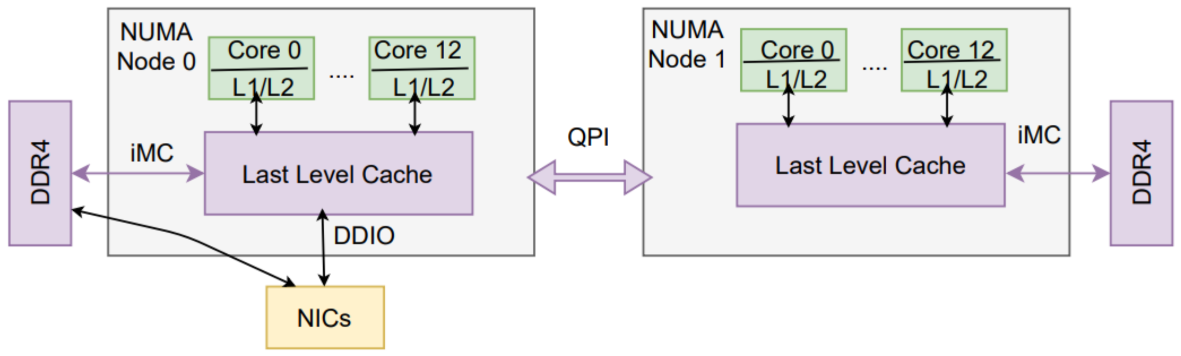


1) Run EPC in one terminal and eNB in another terminal on host1.
2) Run UE in a terminal on host2.
3) Using `iperf3 -s` to start a server on UE using port 5201.
4) Using `iperf3 -c <ip-address>` to start a client on eNB using port 5201.(ip-address of UE)
5) Running 3 iterations of iperf3 for 100 sec each

Test Scenario 2 :

1) Perform flood ping on data sizes of 100, 500,1400,5000, 10000, 15000, 17000 bytes for 1000 packets(`.sudo ping -f -c 30000 -s 500 172.16.0.1` c is count of packets and 500 is the size of packet)
2) In next experiment we didn't keep the number of packets sent fixed, wrote a script that will start and stop ping for x seconds

NUMA Architecture



Non Uniform Memory Access

Non-uniform memory access, or NUMA, is a method of configuring a cluster of microprocessors in a multiprocessing system so they can share memory locally. The idea is to improve the system's performance and allow it to expand as processing needs evolve.

In a NUMA setup, the individual processors in a computing system share local memory and can work together. Data can flow smoothly and quickly since it goes through intermediate memory instead of a main bus.

Factors affecting the throughput :

The first factor is the effect of inter-node resource contention due to cross-node memory access bottleneck.
The second factor is the effect of intra-node resource contention.

Test Scenario 3:

1)Considering all 4 situations of allocating srsepc and srsenb to two nodes :
srsepc on node 0 and srsenb on node 0
srsepc on node 0 and srsenb on node 1
srsepc on node 1 and srsenb on node 0
srsepc on node 1 and srsenb on node 1

Test Scenario 4:

Introducing stress-ng to increase the load on non-allocated cores and doing performance analysis

Test Scenario 5:

Maximizing Total aggregate throughput
Assuming we don’t have sufficient resources to allocated say cores are limited
Statically allocating cores to the functions
Dynamically allocating cores to the functions based on traffic rate.

Performance metrics

a) **Average Sender and Receiver Data Rate**: The data rates can be used to analyze the throughput that is varying based on different values of PRB and MCS. And also throughput is also used to understand how the performance is varying based on switching nodes(CPU) incase of NUMA

b) **SNR:** To understand the channel quality and also the relation between SNR and MCS

Performance Results

Test Scenario 1:

MCS=20 (MCS values are not making a change in data rates)

Running 3 iterations of iperf3 (for 100 sec each)

#PRBS	6	15	25	50	75	100
Sender(1)	3.3	10.3	17.6	35.2	53.0	61.9
Receiver(1)	3.2	10.2	17.4	35	52.8	61.7
Sender(2)	3.4	10.4	17.6	35.2	53	68.6
Receiver(2)	3.5	10.2	17.4	35	52.8	68.4
Sender(3)	3.4	10.4	17.6	35.2	53	58.1
Receiver(3)	3.2	10.2	17.4	35	52.8	57.9

#PRBs	**6**	**15**	**25**	**50**	**75**	**100**
Avg Sender DR achieved (Mbps)	3.32	10.367	17.6	35.2	53	62.867
Avg Receiver DR achieved (Mbps)	3.25	10.2	17.4	35	52.8	62.67

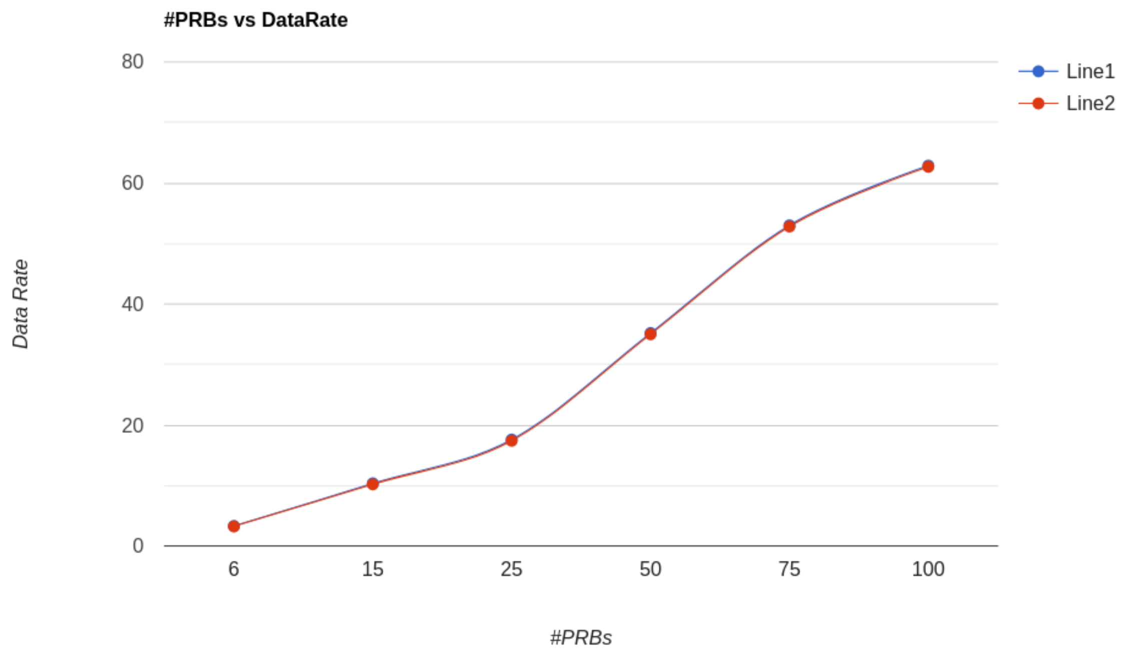


figure1

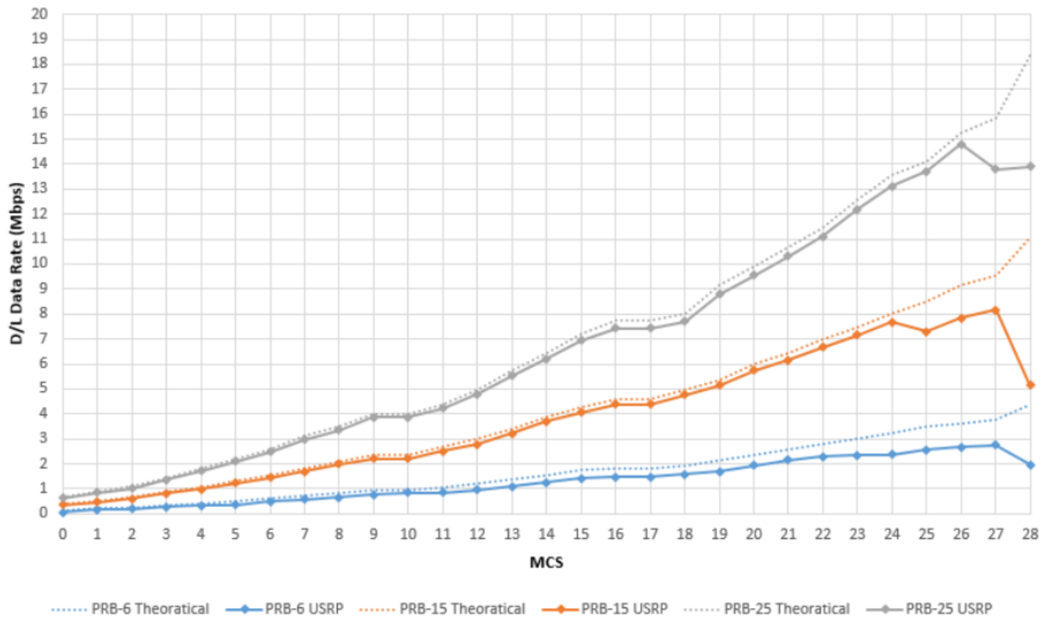


Figure 5.3: UE Downlink Data Rate (USRP) - 1/2

Compared with the results mentioned in **Implementation and performance analysis of software defined radio (SDR) based LTE platform for truck connectivity application** *[References -2]* we didn't include MCS in our plot because though we are changing the MCS in configuration file we are not able to conclude that it is affecting the data rate. So We tried changing PRBs and MCS. However, MCS values were not affecting the BW(higher MCS->higher BW). We took some readings with changing PRBs as follows : (higher PRBs -> higher BW which we were able to observe) and plotted the graph

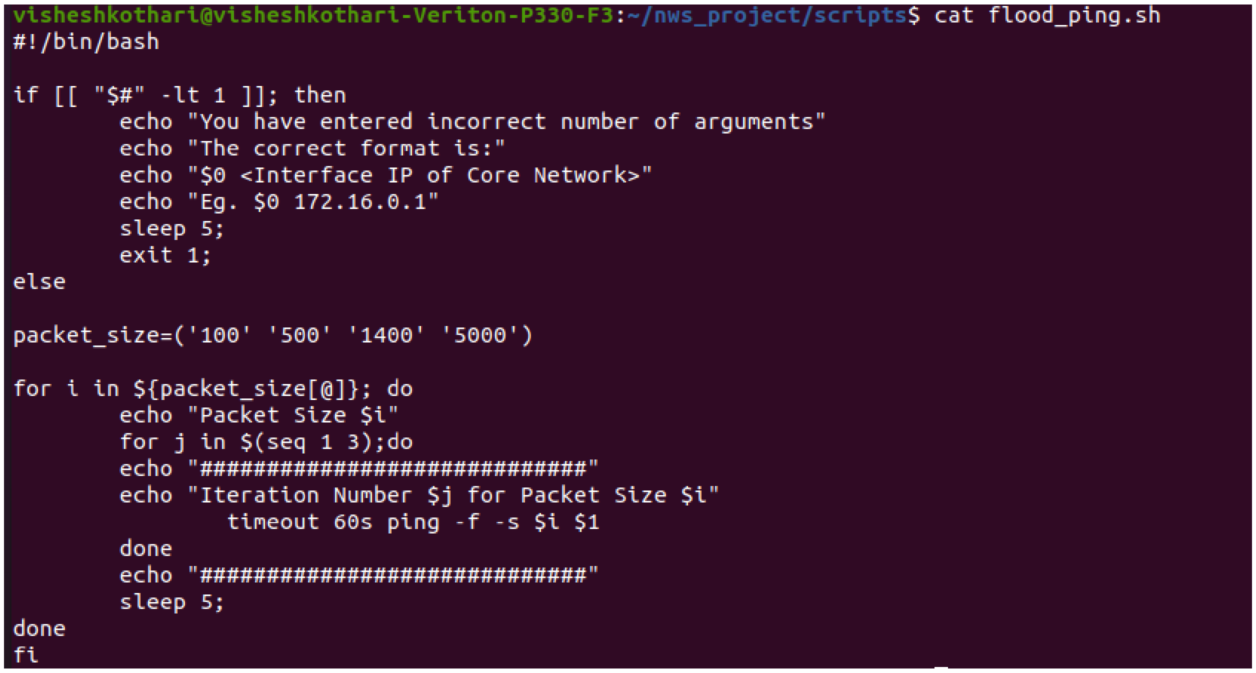
Test Scenario 2

#packets sent	**Size of packet**	**Data size = #packets sent * Size of each packet**	**SNR observed**	**% packet loss**	**Comments**
1,50,000	100	15 MB	34.7848	4	Total time : 2373 sec = 40 min At start, SNR was around 35.7 and after 15-20

					mins, it got reduced till 33.9. Also, after sending 1,19,730 packets, UE attach with ENB failed and packets were not send. Responses received : 1,14,390
30,000	500	15 MB	29.9429	0	468 sec = 7 min
10,000	1500	15 MB	26.8527	17	164 sec = 3 min (8273 packetsreceived)
3,000	5000	15 MB	26.7102	4	48 sec (2851 received)
1,500	10,000	15 MB	24.4892	20	24 sec (1195 received)
1,000	15,000	15 MB	24.8176	31	15 sec (689 received)

From this experiment we were not able to see any significant observations though as packet size is increasing, SNR is decreasing. We can't correlate both as we know SNR is independent of packet size. We later concluded that it's a hardware limitation.

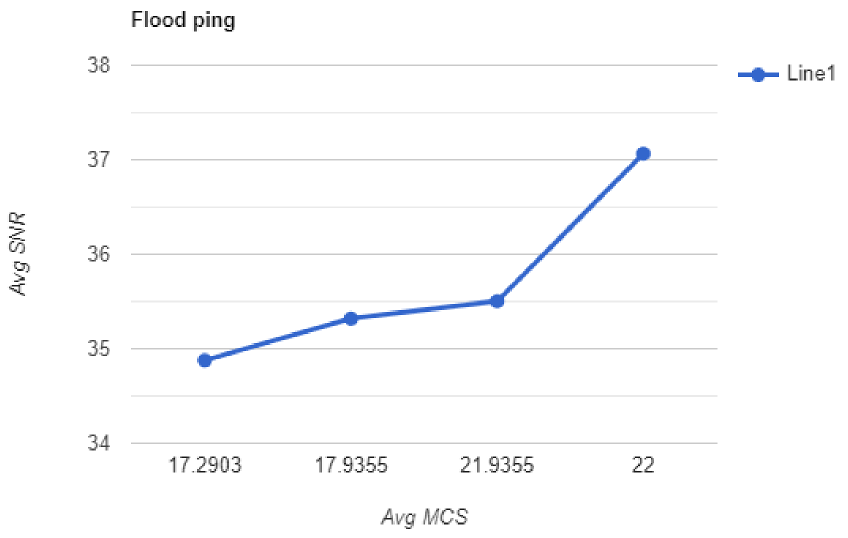
Repeating the sudo flood exp with keeping time constant:

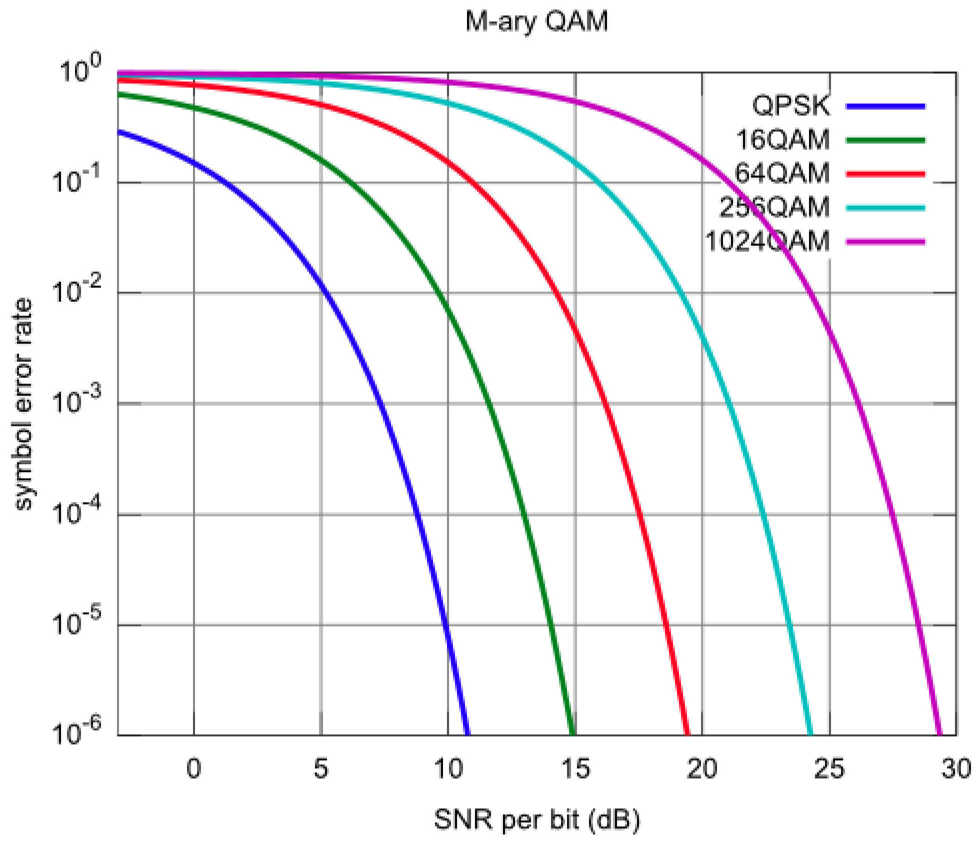


```
visheshkothari@visheshkothari-Veriton-P330-F3:~/nws_project/scripts$ cat flood_ping.sh
#!/bin/bash

if [[ "$#" -lt 1 ]]; then
        echo "You have entered incorrect number of arguments"
        echo "The correct format is:"
        echo "$0 <Interface IP of Core Network>"
        echo "Eg. $0 172.16.0.1"
        sleep 5;
        exit 1;
else

packet_size=('100' '500' '1400' '5000')

for i in ${packet_size[@]}; do
        echo "Packet Size $i"
        for j in $(seq 1 3);do
        echo "##############################"
        echo "Iteration Number $j for Packet Size $i"
                timeout 60s ping -f -s $i $1
        done
        echo "##############################"
        sleep 5;
done
fi
```

Size of packet	**Avg SNR observed**	**Avg MCS observed**
100	37.0645	22
500	34.8774	17.2903
1400	35.3194	17.9355
5000	35.5032	21.9355

From the above values and graph we can observe that for higher SNR values we are getting higher MCS which means from the theoretical perspective as the SNR is decreasing we are switching to lower modulation scheme because the bit Error Rate increases if we keep same modulation scheme for lower SNR
And also higher SNR values leads to switching from lower modulation scheme to higher modulation scheme though bit error rate increases slightly it will increase the data rate.





Test Scenario 3: NUMA without stress-ng
Iteration 1

Same Node	Cores given to EPC	Cores given to ENB	IPC (EPC,ENB)		Sender Avg. Bandwidth	Receiver Avg. Bandwidth
Yes	0-2	3-8	0.71	0.89	17.5	15.8
No	0-2	13-18	0.72	0.87	18.7	17.1
Yes	11-13	14-19	0.74	0.97	17.9	16.2
No	13-15	0-5	0.68	0.94	13.9	12.2

Iteration 2

Same Node	Cores given to EPC	Cores given to ENB	IPC (EPC,ENB)		Sender Avg. Bandwidth	Receiver Avg. Bandwidth
Yes	0-2	3-8	0.79	1.12	15.1	14.9
No	0-2	13-18	0.74	1.01	13.6	13.4
Yes	11-13	14-19	0.77	1.05	13.6	13.4
No	13-15	0-5	0.66	0.98	10	9.87

Iteration 3

Same Node	Cores given to EPC	Cores given to ENB	IPC (EPC,ENB)		Sender Avg. Bandwidth	Receiver Avg. Bandwidth
Yes	0-2	3-8	0.79	1.12	15.1	14.9
No	0-2	13-18	0.74	1.01	13.6	13.4
Yes	11-13	14-19	0.74	0.9	10.3	10.1
No	13-15	0-5	0.66	0.98	10	9.87

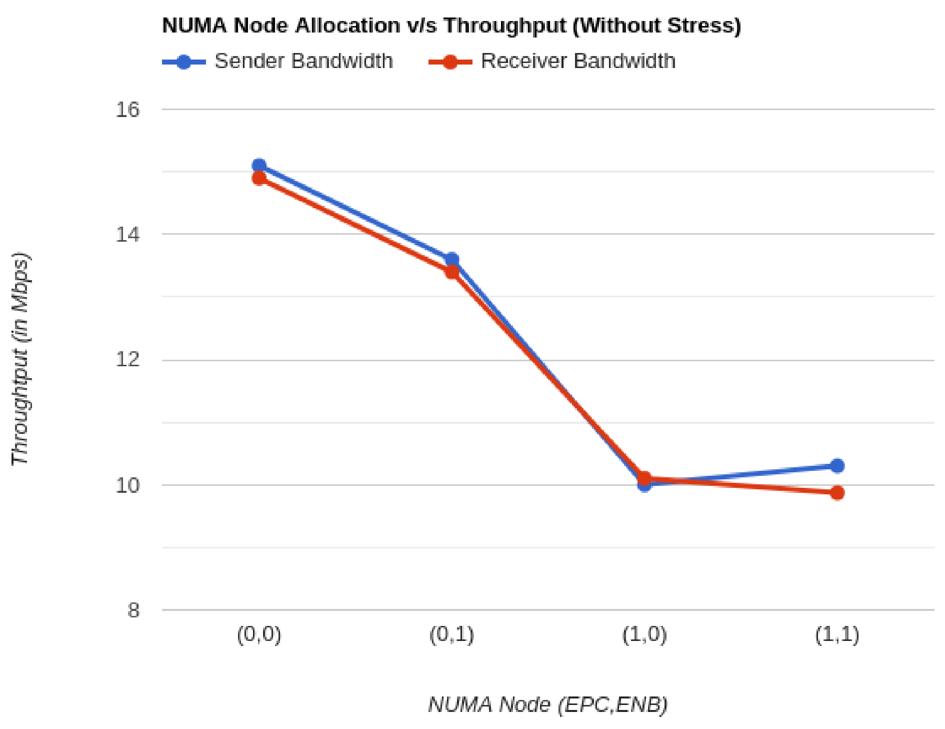


Diagram for Iteration 3

According to the observations mentioned in **"NUMASFP: NUMA-Aware Dynamic Service Function Chain Placement in Multi-Core Servers"[References - 1]** under the section "Performance Impact of SFC Placement in a Server" the maximum throughput is observed when both VNF's are placed on same node that is connected to NIC and low when they are placed on different nodes and as per our readings

In iteration1 :

The values are minimum when EPC is on node 2 and eNB on node 1 (as said low throughput on different nodes) but the maximum throughput is not observed in the case where both EPC and eNB are placed on the same node.

In iteration 2:

The values are showing that maximum throughput is observed incase of both EPC and eNB on same node 0 which is connected to NIC and this results as expected .

The minimum throughput is also observed when the two EPC and eNB are on two different nodes.

In iteration 3:

The results are similar to the analysis of iteration 2.

Test Scenario 4 : NUMA with stress-ng

Iteration 1

Same Node	Cores given to EPC	Cores given to ENB	IPC (EPC,ENB)		Sender Avg. Bandwidth(Mbps)	Receiver Avg. Bandwidth
Yes	0-2	3-8	0.82	1.01	7.15	6.98
No	0-2	13-18	0.82	1.07	7.34	7.15
Yes	11-13	14-19	0.85	1.1	17.7	17.5
No	13-15	0-5	0.84	1.16	11.8	11.7

Iteration 2

Same Node	Cores given to EPC	Cores given to ENB	IPC (EPC,ENB)		Sender Avg. Bandwidth(Mbps)	Receiver Avg. Bandwidth
Yes	0-2	3-8	0.81	1.04	13.3	13.2
No	0-2	13-18	0.79	0.96	12.4	12.2
Yes	11-13	14-19	0.76	0.97	12	11.9
No	13-15	0-5	0.84	1.16	14.6	14.5

We introduced stress-ng to increase the load so that the availability of the cpu time is less for EPC and eNB because the increased load will cause decrease in throughput of EPC and eNB.

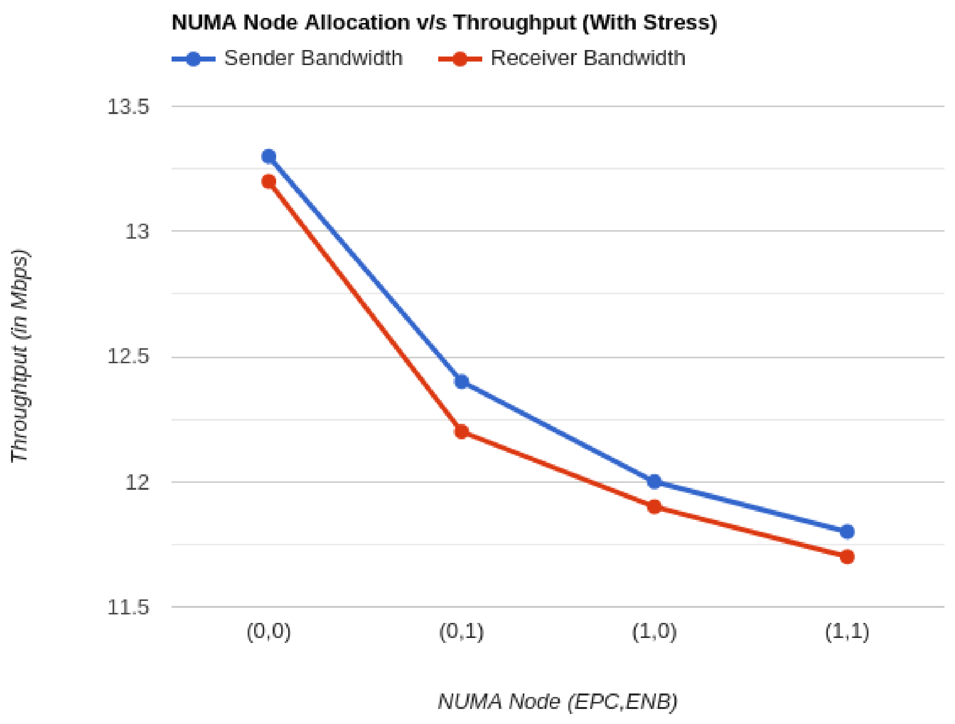


Diagram for iteration 2

Test Scenario 5

Expected Results are that when we allocate cores statically to the functions we don't observe maximum aggregate throughput because consider a scenario where there two functions both are allocated say x and y cores , initially when there is moderate traffic rate on both functions this will not affect throughput much but when the traffic rate on one function is less and needs less than x cores , traffic rate on another function is high and requires more than y cores in this case throughput will be less and the throughput can be increased by dynamically allocating the cores for maximum utilization.

Conclusion

- **Setting up srsLTE allowed us to understand how services like EPC, EnodeB are deployed as VNF and how they need to be managed and allocated resources.**
- **We got insights to important parameters like throughput, SNR, MCS and their inter-relations.**
- **We sometimes do not get the expected results in practice as visualized in theory. We should check and analyze for various factors which may be responsible for the same.**

References

NUMASFP: NUMA-Aware Dynamic Service Function Chain Placement in Multi-Core Servers
Venkatarami Reddy Chintapalli, Sai Balaram Korrapati, Bheemarjuna Reddy Tamma, Antony Franklin A
(2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS))

Contributions

Mahanth Kumar Valluri CS22MTECH11015: Documentation and partial work in implementation and performing experiments.
Vishesh Kothari CS22MTECH12004: Implementation and Performing Experiments and partial work in documentation.
Tejas Deshmukh CS22MTECH12005: Implementation and Performing Experiments and partial work in documentation.

Anti-Plagiarism Statement

I certify that this assignment/report is our own work, based on our personal study and/or research on our personal/lab equipment and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand our responsibility to report honor violations by other students if we become aware of it.

Mahanth Kumar Valluri CS22MTECH11015
Vishesh Kothari CS22MTECH12004
Tejas Deshmukh CS22MTECH12005