OPTIMAL ALLOCATION OF 5G RESOURCES USING REINFORCEMENT LEARNING

BY

Jonnakuti V V M Sri Datta Santosh (19/11/EC/043)

Danam Yashwanth Kumar (19/11/EC/040)

L S Yaswanth Kumar (19/11/EC/059)

Chokkari Dinesh (19/11/EC/062)

Dommara Praveen Kumar (19/11/EC/010)

under the guidance of

Prof. Mukesh Giluka, School of Engineering JNU, Delhi

in the partial fulfilment of the requirements

for the award of the degree of

Bachelor of Technology

(a part of Five-Year Dual Degree Course)

School of Engineering

Jawaharlal Nehru University, Delhi

Feb 2023

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

DECLARATION

We declare that the project work entitled "**OPTIMAL ALLOCATION OF 5G RESOURCES USING REINFORCEMENT LEARNING**" which is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. (a part of Dual-Degree Programme) to School of Engineering, Jawaharlal Nehru University, Delhi comprises only our original work and due acknowledgement has been made in the text to all other material used.

(Full Name and Sign of all Group Members)

Name	**Enrolment No.**	**Signature**
Jonnakuti V V M Sri Datta Santosh	19/11/EC/043	J. Santosh
Danam Yashwanth Kumar	19/11/EC/040	D. Yashwanth
L S Yashwanth Kumar	19/11/EC/059	L.S. Yashwanth
Chokkari Dinesh	19/11/EC/062	[illegible]
Dommara Praveen Kumar	19/11/EC/010	[illegible]

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"OPTIMAL ALLOCATION OF 5G RESOURCES USING REINFORCEMENT LEARNING"** being submitted by **Mr.** Jonnakuti V V M Sri Datta Santosh (Enrolment No. 19/11/EC/043), **Mr.** Danam Yashwanth Kumar (Enrolment No. 19/11/EC/040)**, Mr.** L S Yashwanth Kumar (Enrolment No. 19/11/EC/059), **Mr.** Chokkari Dinesh (Enrolment No. 19/11/EC/062), and **Mr.** Dommara Praveen Kumar (Enrolment No. 19/11/EC/010) in fulfilment of the requirements for the award of the **Bachelor of Technology** (part of Five-Year Dual Degree Course) in **Computer Science Engineering**, will be carried out by them under my supervision.

In my opinion, this work fulfills all the requirements of an Engineering Degree in respective stream as per the regulations of the School of Engineering, Jawaharlal Nehru University, Delhi. This thesis does not contain any work, which has been previously submitted for the award of any other degree.

Dr. Mukesh Giluka
(Supervisor)
Assistant Professor
School of Engineering
Jawaharlal Nehru University, Delhi

JAWAHARLAL NEHRU UNIVERSITY
SCHOOL OF ENGINEERING

ACKNOWLEDGMENT

We would like to take this opportunity to acknowledge everyone who has helped us in every stage of this project.

We are deeply indebted to our mentor Dr. Mukesh Giluka for his guidance and suggestions in completing this project. The completion of this project wouldn't be possible without him.

(Full Name and Sign of all Group Members)

Name	**Enrolment No.**	**Signature**
Jonnakuti V V M Sri Datta Santosh	19/11/EC/043	J. Santosh
Danam Yashwanth Kumar	19/11/EC/040	D. Yashwanth
L S Yashwanth Kumar	19/11/EC/059	L.S. Yashwanth
Chokkari Dinesh	19/11/EC/062	[illegible]
Dommara Praveen Kumar	19/11/EC/010	[illegible]

LIST OF CONTENTS

ABSTRACT

The rapid growth of data traffic has pushed the mobile telecommunication industry towards the adoption of fifth generation (5G) communications. In this project our main aim is to ensure the optimal resource allocation.We have set up a simulation in ns-3 for a 5G NR network using the steps such as configuring the simulation parameters, creating the network topology, configuring the radio access network and defining the traffic patterns.

After configuring all the necessary parameters, we can run the simulation then.We have implemented TcpRlTimeBased protocol using Reinforcement Learning in 5G NR.It applies the RL algorithm to TCP congestion control, for real-time changes in the environment of network transmission, by strengthening the learning management sliding window and threshold size, the network can get better throughputs and smaller delays.

The proposed algorithm utilizes a deep Q-network to learn the optimal congestion window size based on current network conditions and adaptively adjust it in real-time. We evaluate the proposed algorithm using real-world network traces and show that it outperforms the traditional TCP congestion control algorithms.

Additionally, we compare our algorithm with other TCP-based approaches and demonstrate that our method achieves better performance with time. The results demonstrate the effectiveness of the proposed RL-based congestion control algorithm for 500 seconds, which can enhance the overall network performance and improve the user experience.

To Compare the performance of our TcpRlTimeBased protocol with other protocols we have considered the following metrics: Average BSR per packet, End-to-End Delay, Packet Delivery Ratio, RLC-to-RLC Delay, Percentage of SR.

1. INTRODUCTION AND PROJECT OVERVIEW

1.1. INTRODUCTION

In this project we have implemented a new generation TCP protocol TcpRlTimeBased using Reinforcement Learning in 5G NR. It applies the RL algorithm to TCP congestion control, for real-time changes in the environment of network transmission, by strengthening the learning management sliding window and threshold size, the network can get better throughputs and smaller delays.

This protocol takes CWND, SSTHRESH (slow start threshold), segment size, bytes in flight and segments Acked as input and predicts the new CWND and SSTHRESH .

This protocol uses DQN RL Algorithm to predict the CWND size and SSTHRESH. A DQN, or Deep Q-Network, approximates a state-value function in a Q-Learning framework with a neural network.

We have used an ns-3 simulator to setup the 5G environment and simulate this setup. We had 30 UE's, 1 remote host and 1 gNodeB in our setup. Here UE's will be sending packets to remoteHost using TCP Protocol.

Along with TCP RlTimeBased we have simulated the entire scenario for 3 other TCP protocols TcpCubic, TcpIllinois and TCPNewReno to compare the performance of our TCPRlTimeBased protocol.

1.2. PROJECT OBJECTIVE

The objective of the project is to design resource allocation algorithms in 5G with the help of Reinforcement Learning. The designed algorithms will be simulated in NS3 network simulator and performance will be evaluated.

2. LITERATURE SURVEY

2.1. TCP

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

2.1.1. Working of TCP

The TCP/IP model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go.

After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.

For example, When a user requests a web page on the internet, somewhere in the world, the server processes that request and sends back an HTML Page to that user. The server makes use of a protocol called the HTTP Protocol. The HTTP then requests the TCP layer to set the required connection and send the HTML file.

Now, the TCP breaks the data into small packets and forwards it toward the Internet Protocol (IP) layer. The packets are then sent to the destination through different routes.

The TCP layer in the user's system waits for the transmission to get finished and acknowledges once all packets have been received.

2.1.2. Congestion

A state occurs in the network layer when the message traffic is so heavy that it slows down network response time. As delay increases, performance decreases and may lead to retransmission, thus making the situation worse. **Congestion-Avoidance Algorithms (CAA)** are implemented at the TCP layer as the mechanism to avoid congestive collapse in a network.

When congestion occurs, TCP uses a mechanism called Congestion Control to regulate the rate at which data is sent over the network. The goal of congestion control is to prevent the network from becoming overloaded by reducing the sending rate of the data. This is done by reducing the TCP window size, which limits the amount of data that can be sent before an acknowledgment is received.

TCP uses a feedback mechanism to detect congestion. When a packet is lost, TCP assumes that congestion has occurred and reduces the sending rate. TCP also monitors the round-trip time (RTT) for packets, and if the RTT increases beyond a certain threshold, TCP assumes that congestion has occurred and reduces the sending rate.

2.1.3. TCP Congestion Control

TCP uses a congestion window and a congestion policy that avoids congestion. Previously, we assumed that only the receiver could dictate the sender's window size. We ignored another entity here, the network. If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

TCP congestion control is an important part of network management, as it helps to prevent network congestion and ensure reliable data transmission. Different congestion control algorithms are used by different TCP implementations, including Reno, New Reno, and CUBIC.

Congestion policy in TCP -

1. Slow Start Phase: starts slowly increment is exponential to threshold
2. Congestion Avoidance Phase: After reaching the threshold increment is by 1.
3. Congestion Detection Phase: Sender goes back to slow start phase or congestion avoidance phase.

2.2. TCP VARIANTS

2.2.1. TCP New Reno

TCP New Reno is the extension of TCP Reno. It overcomes the limitations of Reno. TCP Reno is the second variant of the TCP which came up with an in-built congestion algorithm. Congestion handling was not an integral part of the original TCP/IP suite. TCP Reno is the extension of TCP Tahoe, and NewReno is the extension of TCP Reno. In Reno, when packet loss occurs, the sender reduces the CWND by 50% along with the SSTHRESH value. This would allow the network to come out of the congestion state easily. But Reno suffered from a very critical backlog which hurts its performance.

It uses the concept of partial acknowledgement. When the sender receives the ACK of the first retransmitted packet then it does not consider it a “New ACK” unlike TCP Reno. NewReno checks if all the previously transmitted packets of that particular window are ACKed or not. If there are multiple packets lost in the same congestion window then the receiver would have been sending the duplicate ACKs only even after receiving the retransmitted packet. This will make it clear to the sender that all the packets have not reached the receiver and hence the sender will not consider that ACK as new. It will consider it a partial ACK because only a partial window is being ACKed not the whole. Reno used to come out of the fast recovery phase after receiving a new ACK, but NewReno considers that ACK as partial and does not come out of the fast recovery phase. It wisely makes the decision of ending the fast recovery phase when it receives the Cumulative ACK of the entire congestion window.

2.2.2 TCP Cubic

TCP CUBIC is a congestion control algorithm that arises with the idea of taking advantage of the fact that today’s communications links tend to have increasingly higher bandwidth levels.
In a network composed of wide bandwidth links, a congestion control algorithm that slowly increases the transmission rate may end up wasting the capacity of the links.
The intention is to have an algorithm that works with congestion windows whose incremental processes are more aggressive, but are restricted from overloading the network.
In order to achieve this, it is proposed that the scheme for increasing and decreasing the transmission ratio be established according to a cubic function.

The procedure followed by the algorithm is:

1. At the time of experiencing congestion event the window size for that instant will be recorded as Wmax or the maximum window size.
2. The Wmax value will be set as the inflection point of the cubic function that will govern the growth of the congestion window.
3. The transmission will then be restarted with a smaller window value and, if no congestion is experienced, this value will increase according to the concave portion of the cubic function.
4. As the window approaches Wmax the increments will slow down.
5. Once the tipping point has been reached, i.e. Wmax, the value of the window will continue to increase discreetly.
6. Finally, if the network is still not experiencing any congestion, the window size will continue to increase according to the convex portion of the function.

2.2.3. TCP Illinois

It is based on the idea of measuring the round-trip time (RTT) of packets in the network to estimate the level of congestion. TCP Illinois uses a nonlinear function to adjust the congestion window size based on the estimated level of congestion. This allows it to achieve higher throughput than other congestion control algorithms, while still maintaining low packet loss rates.

TCP Illinois is designed to be compatible with existing TCP implementations, so it can be used in networks that already use TCP. It has been shown to be effective in a wide range of network scenarios, including high-speed networks and wireless networks.

TCP Illinois works by measuring the round-trip time (RTT) of packets in the network to estimate the level of congestion. When a TCP connection starts, TCP Illinois sets the initial congestion window size (CWND) to a small value. This is done to avoid overwhelming the network with too much traffic at the beginning of the connection.

As data is transmitted over the connection, TCP Illinois uses the measured RTT to estimate the level of congestion in the network. If the RTT is low, it means that there is little congestion and the CWND can be increased. If the RTT is high, it means that there is congestion and the CWND should be decreased to reduce the amount of traffic in the network.

TCP Illinois uses a nonlinear function to adjust the CWND based on the estimated level of congestion. The function is designed to be more aggressive in increasing the CWND when the network is lightly loaded, and more conservative in decreasing the CWND when the network is congested. This allows TCP Illinois to achieve higher throughput than other congestion control algorithms, while still maintaining low packet loss rates.

In addition, TCP Illinois uses a mechanism called Fast Recovery to quickly recover from packet losses. When a packet is lost, TCP Illinois reduces the CWND by a smaller amount than other algorithms, and then enters a state called Recovery. In this state, TCP Illinois sends new packets at a slower rate to avoid further congestion, while still trying to maintain a high throughput. This allows TCP Illinois to recover quickly from packet losses, while still maintaining good performance.

2.2.4. TCP RlTimeBased

TCP RlTimeBased is a new generation TCPNewReno Protocol. It applies the RL algorithm to TCP congestion control, for real-time changes in the environment of network transmission, by strengthening the learning management sliding window and threshold size, the network can get better throughputs and smaller delays.

This protocol takes CWND, SSTHRESH (slow start threshold), segment size, bytes in flight and segments Acked as input and predicts the new CWND and SSTHRESH .

This protocol uses DQN RL Algorithm to predict the CWND size and SSTHRESH. A DQN, or Deep Q-Network, approximates a state-value function in a Q-Learning framework with a neural network.

2.3. NETWORK SIMULATOR

A discrete-event network simulator called ns-3 is primarily intended for academic and research purposes. Free software under a GNU GPLv2 licence, ns-3 is accessible to the general public for usage, development, and research.

It provides a modular framework for developing network simulations that can model various types of communication networks and protocols. ns-3 supports a wide range of network protocols and applications, including TCP, UDP, HTTP, and routing protocols such as OSPF and BGP.

ns-3 provides a rich set of tools for network simulation, including the ability to generate traffic, configure network topologies, and collect statistics on network performance. It can simulate wireless networks, including both Wi-Fi and cellular networks, as well as wired networks. ns-3 also supports various mobility models, such as random waypoint and random walk models, to simulate mobile devices in a network.

The ns-3 project aims to create a free and open source simulation environment appropriate for networking research. It should be in line with the simulation requirements of contemporary networking research and should promote peer review, community participation, and software validation. An international group of volunteers maintains ns-3.

2.3.1. Ns-3 Installation

Step 1: Open a terminal in your Ubuntu machine or VM

Step 2: Check for G++ version in your Ubuntu OS with following command → **g++ --version**
Output will be as follows:

```
fire@fire-Standard-PC-i440FX-PIIX-1996:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

As highlighted in the previous output, G++ version is 9.4.0. So, it's good to go!!
Otherwise, if (G++ version is older(<) than 8.0.0), get the latest one by running following commands:-

```
sudo apt-get update
sudo apt-get install g++-8.0.0
```

Step 3: From the terminal run the following commands:

```
sudo apt install git-all
git clone https://gitlab.com/nsnam/ns-3-dev.git
cd ns-3-dev
$ git remote add nsnam https://gitlab.com/nsnam/ns-3-dev.git
$ git checkout master
$ git pull nsnam master
```

Step 4: Now build the ns-3 with following command: **sudo apt-get install -y build-essential**
Output will be as follows:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
build-essential set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 78 not upgraded.
```

Step 5: Run cmake command: **sudo apt install cmake**
Output will be as follows:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cmake-data libjsoncpp1 librhash0
Suggested packages:
  cmake-doc ninja-build
```

Step 6: Install required packages with the following commands:

```
sudo apt install g++ python3 python3-dev pkg-config sqlite3 cmake
sudo apt install sqlite sqlite3 libsqlite3-dev
```

Note: These commands are for installing missing packages.

Step 7: Run the following commands:

```
git checkout ns-3.35
./waf configure --build-profile=debug --enable-examples --enable-tests
./waf build
```

Final lines of the output are as follows:

```
-- ---- Summary of optional ns-3 features:
Build profile                    : debug
Build directory                  : /home/fire/Documents/ns-3-dev/build
Build version embedding          : OFF (not requested)
BRITE Integration                : OFF (missing dependency)
DES Metrics event collection  : OFF (not requested)
DPDK NetDevice                   : OFF (not requested)
Emulation FdNetDevice            : ON
Examples                         : ON
File descriptor NetDevice        : ON
GNU Scientific Library (GSL)  : OFF (missing dependency)
GtkConfigStore                   : OFF (missing dependency)
LibXml2 support                  : OFF (missing dependency)
MPI Support                      : OFF (not requested)
ns-3 Click Integration           : OFF (missing dependency)
ns-3 OpenFlow Integration        : OFF (missing dependency)
Netmap emulation FdNetDevice  : OFF (missing dependency)
PyViz visualizer                 : OFF (missing dependency)
Python Bindings                  : OFF (not requested)
SQLite support                   : OFF (missing dependency)
Tap Bridge                       : ON
Tap FdNetDevice                  : ON
Tests                   : ON


Modules configured to be built:
antenna             aodv                applications
bridge              buildings           config-store
core                csma                csma-layout
dsdv                dsr                 energy
fd-net-device       flow-monitor        interne
internet-apps       lr-wpan             lte
mesh                mobility            netanim
network             nix-vector-routing          olsr
point-to-point      point-to-point-layout       propagation
sicslowpan          spectrum            stats
tap-bridge          test                topology-read
traffic-control     uan                 virtual-net-device
wave                wifi                wimax


Modules that cannot be built:
brite               click               mpi
openflow            visualizer


-- Configuring done
-- Generating done
-- Build files have been written to: /home/fire/Documents/ns-3-dev/cmake-cache
Finished executing the following commands:
mkdir cmake-cache
cd cmake-cache; /usr/bin/cmake -DCMAKE_BUILD_TYPE=debug -DNS3_ASSERT=ON -DNS3_LOG=ON
-DNS3_WARNINGS_AS_ERRORS=ON -DNS3_NATIVE_OPTIMIZATIONS=OFF -DNS3_EXAMPLES=ON
-DNS3_TESTS=ON -G Unix Makefiles .. ; cd ..
```

Following these above mentioned steps will result in successful installation of NS3 on your machine and you’re good to go.
There are example codes in NS3 directory, you can run them with the following commands
./waf --run first
Note: first is <filename>.
Output is as follows:

```
fire@fire-Standard-PC-i440FX-PIIX-1996:~/Documents/ns-3-dev$ ./ns3 run first
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

2.4. NS-3-AI

Currently, more and more researchers are willing to apply AI algorithms to network research. Most AI algorithms are likely to rely on open source frameworks such as TensorFlow and PyTorch, but these frameworks are developed independently of ns-3 and extremely hard to merge, so it is more reasonable and convenient to connect them with data interaction. This ns-3 extension module provides a high-efficiency solution to enable the data interaction between ns-3 and other python based AI frameworks.

Inspired by ns3-gym, but using a different approach (shared memory) which is faster and more flexible

2.4.1 Working of NS-3-AI

The ns3-ai module consists of two components, the ns-3 interface developed by C++ and the AI interface developed by Python. This module provides a high-level interface for the quick development of DL/RL algorithms as well as the core module to transfer data from one C++ program to another Python program.

The key concept of interaction between AI frameworks and ns-3 is to enable data to transfer in multiple processes. Each process has a different user address space, and the global variables of any process cannot be seen in the other process, so to exchange data between processes must pass through the buffer in the ker-nel (e.g., process A puts data from user space to the kernel buffer, process B reads the data from the kernel buffer).

Several ways can be adapted to the communication between processes such as pipe, socket (used by ns3-gym), and shared memory. In the next generation of wireless communication networks, the density and complexity of networks are rising rapidly, leading to an increase in data and training time. Thus, the ability to exchange a large volume of data in a short time should be considered first, that is why shared memory is chosen as the core module to transfer data.

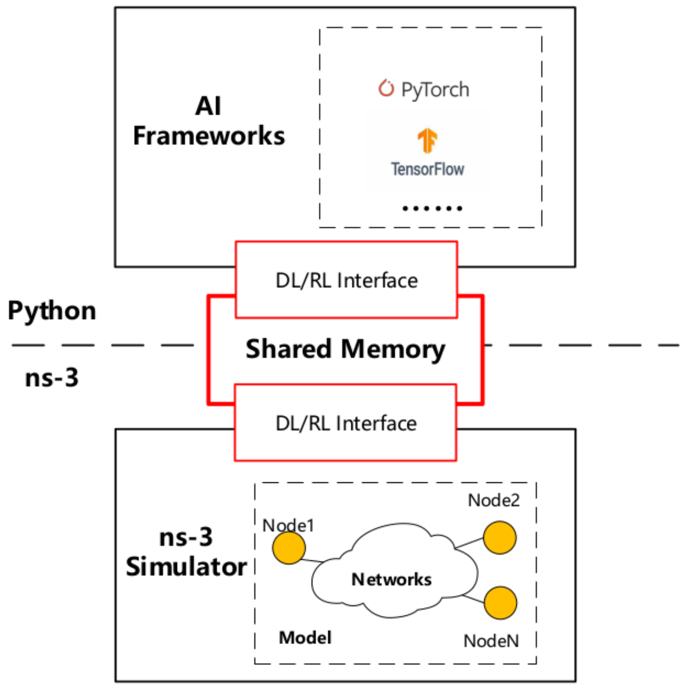

Fig 1. System Architecture of NS-3-AI

2.4.2. NS-3-AI Installation

Step 1: Clone ns3-ai repository in contrib directory with the following commands

```
cd ./contrib
git clone https://github.com/hudt/diangroup/ns3-ai
```

Step 2: Configure and Build ns3 project with the following commands

```
./waf configure
./waf build
```

Step 3: Add Python Interface with the following commands

```
cd ./contrib/ns-3-ai/py_interface
pip3 install . --user
```

Step 4: Run examples with the following commands

```
cp -r contrib/ns3-ai/examples/a_plus_b_scratch
cd scratch/a_plus_b/
python3 run.py
```

2.5. 5G NR

5G NR (New Radio) is a standard for wireless communication that defines the air interface for 5G networks. It is part of the 5G mobile network technology that promises to offer higher data rates, lower latency, better energy efficiency, and more reliable communication than previous cellular networks.

5G NR is designed to operate in a wide range of frequency bands, including low, mid, and high bands. It uses advanced modulation techniques such as QAM and OFDM to achieve high data rates and spectral efficiency. 5G NR also introduces new features such as beamforming, network slicing, and support for massive MIMO (Multiple Input Multiple Output) to improve the overall network performance and capacity.

One of the key features of 5G NR is its ability to support a wide range of use cases, including enhanced mobile broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable and low-latency communications (URLLC). This allows 5G networks to support a diverse range of applications, such as virtual reality, autonomous vehicles, and the Internet of Things (IoT).

5G NR is backward compatible with previous cellular network technologies, such as 4G LTE, which means that it can coexist and interoperate with existing networks. This allows for a smooth transition to 5G without the need for a complete overhaul of the existing infrastructure. Overall, 5G NR is a promising technology that is expected to revolutionize the way we connect and communicate in the future, enabling a wide range of new applications and services that were not possible before.

2.5.1. 5G NR Installation

Step-1: In ns-3-dev/src directory clone nr repository with the following commands

```
git clone https://gitlab.com/cttc-lena/nr.git
```

Step-2: ./waf configure --enable-examples --enable-tests

Step-3: ./waf build -j10

Now you can run examples from 5G NR as well, run one of the examples with following command

```
./waf --run cttc-3gpp-channel-nums
```

Output will be as follows:

```
fire@fire-Standard-PC-i440FX-PIIX-1996:~/Documents/ns-3-dev$ ./waf --run cttc-3gpp-channel-nums
Waf: Entering directory `/home/fire/Documents/ns-3-dev/build'
Waf: Leaving directory `/home/fire/Documents/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.963s)

 Total UDP throughput (bps):3.72507e+08
```

2.5.2. 5G NR Uplink

In 5G NR (New Radio), the uplink is the direction of data transmission from the User Equipment (UE) to the base station. The uplink in 5G NR is designed to support higher data rates, lower latency, and improved efficiency compared to previous generation networks.

This is achieved through a number of key technologies and features, including:

1. Multiple-Input Multiple-Output (MIMO): 5G NR uses advanced MIMO techniques to increase the capacity and spectral efficiency of the uplink. By using multiple antennas at both the UE and the base station, 5G NR can transmit and receive multiple streams of data simultaneously, increasing the overall throughput.
2. Higher bandwidth: 5G NR supports higher uplink bandwidths compared to previous generation networks, enabling faster data transfer rates and reduced latency.
3. Flexible numerology: 5G NR supports a flexible numerology, which allows the transmission of data in different subcarrier spacings and symbol durations. This enables more efficient use of the available spectrum, especially in low to medium frequency bands.
4. Grant-free access: In some scenarios, such as IoT devices with low data rate requirements, 5G NR supports grant-free access, which allows UEs to transmit data without waiting for a grant from the base station. This reduces the latency and improves the efficiency of uplink transmissions.
5. Dynamic scheduling: 5G NR supports dynamic scheduling, which enables the base station to allocate uplink resources to UEs based on their traffic demands and channel conditions.

This enables more efficient use of the available resources, especially in highly dynamic and diverse network scenarios.

2.5.2. 5G NR downlink

In 5G NR (New Radio), the downlink is the direction of data transmission from the base station to the User Equipment (UE). The downlink in 5G NR is designed to support higher data rates, lower latency, and improved efficiency compared to previous generation networks.

This is achieved through a number of key technologies and features, including:

1. Beamforming: 5G NR uses advanced beamforming techniques to improve the signal quality and coverage of the downlink. By focusing the downlink signal in the direction of the UE, beamforming enables higher data rates and improved spectral efficiency.
2. Higher bandwidth: 5G NR supports higher downlink bandwidths compared to previous generation networks, enabling faster data transfer rates and reduced latency.
3. Flexible numerology: 5G NR supports a flexible numerology, which allows the transmission of data in different subcarrier spacings and symbol durations. This enables more efficient use of the available spectrum, especially in low to medium frequency bands.
4. Massive MIMO: 5G NR supports massive MIMO, which uses a large number of antennas at the base station to transmit multiple streams of data simultaneously to multiple UEs. This enables higher data rates and improved spectral efficiency, especially in dense urban environments.
5. Dynamic scheduling: 5G NR supports dynamic scheduling, which enables the base station to allocate downlink resources to UEs based on their traffic demands and channel conditions. This enables more efficient use of the available resources, especially in highly dynamic and diverse network scenarios.

3. PROPOSED WORK AND METHODOLOGY

3.1 SETUP

To set up a simulation in ns-3 for a 5G NR network with 10 UEs and 1 gNodeB, we should need to perform the following steps:

1. Install the necessary modules: You would need to install the ns-3 software along with the 5G NR module for ns-3. The 5G NR module provides the necessary functionality for simulating a 5G network.
2. Configure the simulation parameters: We should need to set the simulation parameters, such as the simulation time, the number of UEs and gNodeBs, the frequency band, channel bandwidth, and other network parameters.
3. Create the network topology: We should need to create the network topology by defining the position and mobility of the UEs and the gNodeB. You can use existing mobility models or create your own custom model.
4. Configure the radio access network: We should need to configure the radio access network by setting the appropriate MAC, PHY, and RLC parameters for the UEs and gNodeB. You can also set the transport network parameters such as the number of bearers and their quality of service (QoS) parameters.
5. Define the traffic patterns: You would need to define the traffic patterns for the UEs by setting the data rates, packet sizes, and inter-packet intervals. You can use existing traffic models or create your own custom model.

6. Run the simulation: Once we have configured all the necessary parameters, you can run the simulation. During the simulation, ns-3 will generate log files and output statistics that can be used to evaluate the performance of the network.

3.1.1. NS-3 Simulation Setup

3.1.1.1 NR Helper

- ➔ In ns-3, the nrHelper object is a helper class that simplifies the process of setting up an NR (New Radio) network simulation.
- ➔ Create the nrHelper Object to assign streams, to attach uE to eNodeB

```
// Create an instance of nrHelper
Ptr<NrHelper> nrHelper = CreateObject<NrHelper>      ();
```

- ➔ Create a Spectrum Division band and Initialize the Operation Band
- ➔ With the help of nrHelper set gNodeB attributes such as Transmission Power, Numerology, gNodeB, and uE Antenna Attributes.
- ➔ Set all the above values to default values.
- ➔ The EPC (Evolved Packet Core) Helper in ns-3 is a module that helps to set up an LTE and 5G NR network simulation. The EPC is a key component of the LTE and 5G NR network architecture, and it provides services such as mobility management, session management, and packet routing.
- ➔ In a 5G NR simulation, the EPC Helper can be used to create an EPC network that consists of multiple components such as the MME, SGW, and PGW. The EPC Helper provides methods for setting up and configuring these components, as well as for connecting them to the 5G NR gNodeB devices and UE devices in the simulation.
- ➔ Create an epc helper to implement internet on gNodeB

```
// Create an instance of epcHelper
Ptr<NrPointToPointEpcHelper> epcHelper =
CreateObject<NrPointToPointEpcHelper> ();
```

- ➔ Create 30 uE and 1 gNodeB and fix the Mobility with the help of MobilityHelper Object.
- ➔ In this setup gNodeB is stationary and remains the same throughout the simulation. Initially the UE's are placed on a circle and are tasked to perform the random walk during the simulation.

3.1.1.2 Numerology and Transmission Power

Numerology in 5G NR refers to the set of parameters that define the frequency, time, and subcarrier spacing of the signals used in the air interface of 5G networks. Different numerologies are used for different use cases, and the choice of numerology depends on the specific requirements of the network.

Lower numerologies are more suited for applications with low mobility, such as fixed wireless access, while higher numerologies are better suited for high mobility applications, such as connected vehicles.

```
nrHelper->SetGnbPhyAttribute ("Numerology", UintegerValue (numerology));
```

The transmission power is a critical parameter in wireless communications as it determines the coverage area and the signal strength at the receiver.

```
nrHelper->SetGnbPhyAttribute ("TxPower", DoubleValue (txPower));
```

3.1.1.3 Antenna and NrMacScheduler

Antennas are responsible for transmitting and receiving signals between base stations (gNodeBs) and user equipment (UEs). In 5G NR, there are several types of antennas that can be used depending on the specific use case and deployment scenario.In this setup we have used IsotropicAntenna.

IsotropicAntennaModel is a simple and commonly used antenna model that assumes the antenna radiates or receives signals equally in all directions. The IsotropicAntennaModel is a special case of an omnidirectional antenna.

```
nrHelper->SetUeAntennaAttribute ("AntennaElement", PointerValue
(CreateObject<IsotropicAntennaModel> ()));

nrHelper->SetGnbAntennaAttribute ("AntennaElement", PointerValue
(CreateObject<IsotropicAntennaModel> ()));
```

The MAC (Medium Access Control) layer is responsible for managing the communication between the gNodeB and the UE. One of the key components of the MAC layer is the MAC scheduler, which is responsible for deciding how and when to allocate the available radio resources (time-frequency blocks) to different UEs in the network.The MAC scheduler uses various algorithms to make these decisions, with the goal of optimizing the use of the available resources to meet the specific requirements of the UEs and the network.

The MAC scheduler can operate in different scheduling modes, such as round-robin (RR), proportional fair (PF), and maximum rate (MR).In this setup we have used PF scheduling mode.

In PF mode, the MAC scheduler allocates the available radio resources (time-frequency blocks) to the UEs in a way that maximizes the overall network throughput while maintaining a fair distribution of resources among the UEs.The PF mode algorithm prioritizes UEs that have a higher channel quality while also ensuring that lower quality UEs still receive some resources to maintain a minimum level of service. The basic idea behind PF is to allocate resources in a way that maximizes the "proportional fairness" of the system.

```
nrHelper->SetSchedulerTypeId (TypeId::LookupByName ("ns3::NrMacSchedulerTdmaPF"));
```

3.1.1.4 Constant Position Mobility Model

The Constant Position Mobility Model in ns-3 is a simple mobility model that keeps nodes stationary at a constant position throughout the simulation. This mobility model is useful for testing scenarios in which nodes do not move, such as when analyzing the performance of a fixed network topology.

To use the Constant Position Mobility Model in ns-3, we need to create an instance of the model and set its parameters.

```
// Create gNodeB Nodes
NodeContainer gNbNodes;

// Mobility Helper to set Mobility of gNodeB
MobilityHelper gNBmobility;

// Position Allocation of gNodeB
Ptr<ListPositionAllocator> bsPositionAlloc = CreateObject<ListPositionAllocator> ();

// gNodeB Height
const double gNbHeight = 10;
gNbNodes.Create (1);

// Position of gNodeB
bsPositionAlloc->Add (Vector (500.0, 500.0, gNbHeight));

// Set Constant Position Mobility Model for gNodeB and install Mobility on gNodeB
gNBmobility.SetPositionAllocator (bsPositionAlloc);
gNBmobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
gNBmobility.Install (gNbNodes);
```

3.1.1.5 Random Walk 2D Mobility Model

The RandomWalk2d model in ns-3 is a mobility model that simulates the movement of entities in a 2D plane using a random walk pattern. The model generates a sequence of random steps, in which each step has a random direction and magnitude, based on certain probability distributions.

To use the RandomWalk2d model in ns-3, we need to first create an instance of the mobility model and set its parameters.

```
// Create UE Nodes
NodeContainer ueNodes;
// Mobility Helper to set Mobility of UE's
MobilityHelper uESMobility;

// Position Allocation of UE's
Ptr<ListPositionAllocator> utPositionAlloc = CreateObject<ListPositionAllocator> ();

// UE Height
const double ueHeight = 1.5;
ueNodes.Create (noOfUEs);

// Position of UE's
float angle = 0.0;
float theta = (360 / noOfUEs)*3.14159/180;
int radius = 499;
for(uint32_t i=0; i<noOfUEs; i++) {
        utPositionAlloc->Add (Vector (radius*(1 + sin(angle)), radius*(1 + cos(angle)),
ueHeight));
      angle += theta;
}

// Set Random Walk Mobility Model for UE's and install    Mobility on UE's
uESMobility.SetPositionAllocator (utPositionAlloc);
uESMobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                         "Mode", StringValue ("Time"),
                         "Time", StringValue ("1s"),
                         "Speed", StringValue ("ns3::UniformRandomVariable[Min=5.0|Max=15.0]"),
                         "Bounds", RectangleValue (Rectangle (0.0, 1000.0, 0.0, 1000.0)));

uESMobility.Install (ueNodes);
```

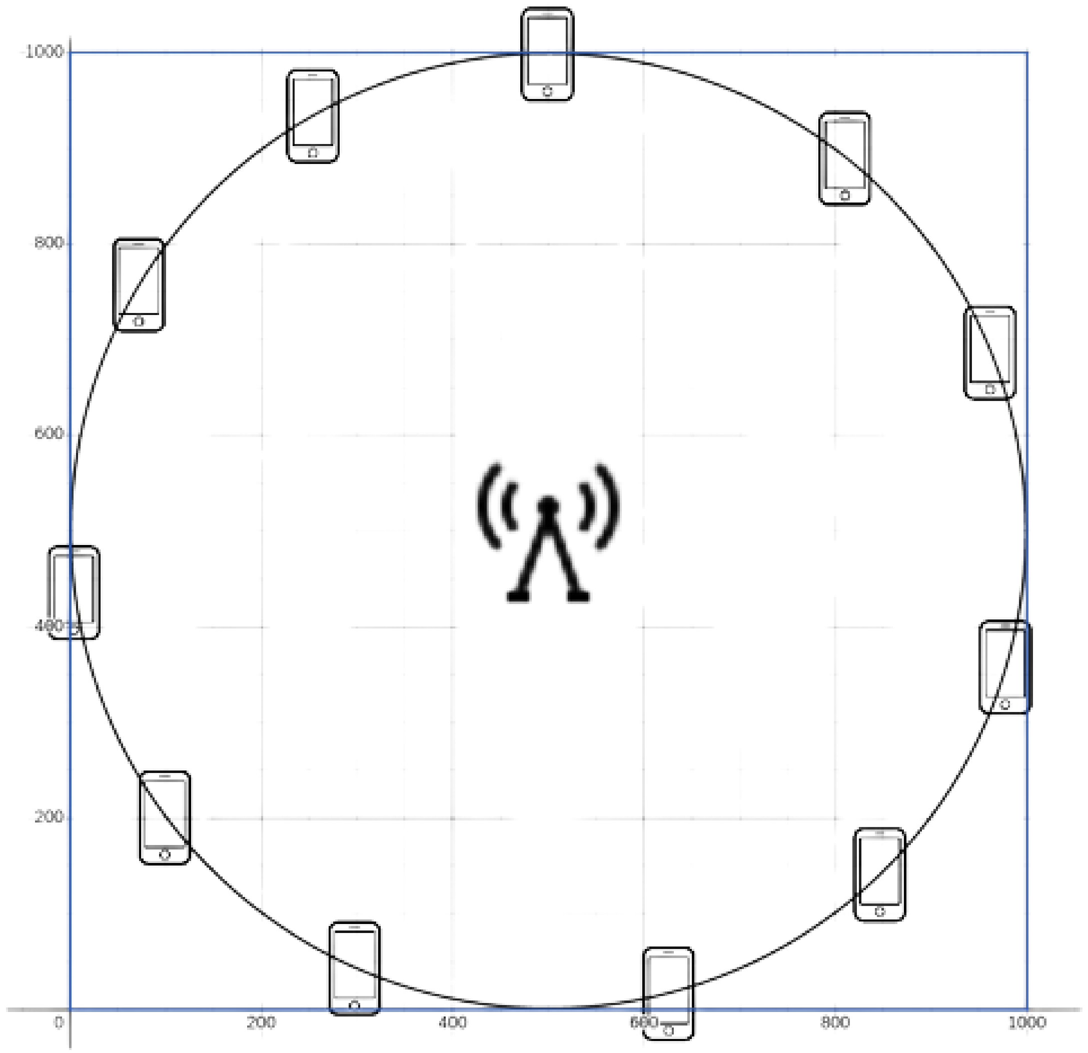

Fig 2. Position of UE's and gNodeB

3.1.1.6 5G NR Network and Routing

➔ Install nr net on uE and gNodeB

```
// To Install nr net on UE's and gNodeB
```

```
NetDeviceContainer uENetDev =
nrHelper->InstallUeDevice(uENodes, allBwps);

NetDeviceContainer gNbNetDev = nrHelper->InstallGnbDevice(eNbNodes,
allBwps);
```

➔ Assign Streams for both UE's and gNodeB

```
int64_t randomStream = 1;
randomStream += nrHelper->AssignStreams(gNbNetDev, random);
randomStream += nrHelper->AssignStreams(uENetDev, random);
```

➔ When All the Configuration is done, explicitly call UpdateConfig()
➔ To Update the Configurations that are set till now.

```
for (auto it = gNbNetDev.Begin (); it != gNbNetDev.End (); ++it)
{
        DynamicCast<NrGnbNetDevice> (*it)->UpdateConfig ();
}

for (auto it = ueNetDev.Begin (); it != ueNetDev.End (); ++it)
{
    DynamicCast<NrUeNetDevice> (*it)->UpdateConfig ();
}
```

➔ In 5G NR network architecture, the PGW is a key component of the EPC network. Its main purpose is to provide a gateway between the 5G NR network and external packet data networks, such as the internet.
➔ When a UE (user equipment) in a 5G NR network wants to access an external packet data network, such as a website on the internet, it sends a request to the gNB (base station) through the 5G NR air interface. The gNB then forwards the request to the PGW through the S1-U interface of the EPC network.
➔ Create the Internet and Install IP Stack on UE's and Get PGW and Create a Single RemoteHost

```
Ptr<Node> pgw = epcHelper->GetPgwNode ();
// Create a remoteHost
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
// To Install internet on remoteHost
InternetStackHelper internet;
internet.Install (remoteHostContainer);
```

➔ Connect RemoteHost to PGW and Setup Routing

```
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", StringValue (bottleneck_bandwidth));
p2ph.SetChannelAttribute ("Delay", StringValue (bottleneck_delay));

// Connecting remoteHost to PGW
NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);

Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"),
Ipv4Mask ("255.0.0.0"), 1);
  internet.Install (ueNodes);
```

3.1.1.7 Assignment of IP Address

➔ Assign IP Address for uE

```
// To Assign IP Addresses for UE's
Ipv4InterfaceContainer ueIpIface = epcHelper->AssignUeIpv4Address
(NetDeviceContainer (ueNetDev));
```

➔ Set Default Gateway for uE

```
Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting
(ueNodes.Get (0)->GetObject<Ipv4> ());
  ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (),
1);
```

3.1.1.8 UE and gNodeB Connection

➔ Attach UE's to Closest eNodeB

```
// attach UE's to the closest eNB
nrHelper->AttachToClosestEnb (ueNetDev, gNbNetDev);
```

3.1.1.9 Sink and Source Applications

➔ Install and Start Sink Applications

```
PacketSinkHelper sink ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), 9));
```

```
ApplicationContainer sinkApps = sink.Install (remoteHost);
sinkApps.Start (Seconds (0.0));
sinkApps.Stop (Seconds (49.9));
```

➔ Install and Start Source Applications

```
BulkSendHelper source ("ns3::TcpSocketFactory", InetSocketAddress (Ipv4Address
("7.0.0.2"), 9));
source.SetAttribute ("SendSize", UintegerValue (1000));
ApplicationContainer sourceApps = source.Install (ueNodes.Get (i));
sourceApps.Start (Seconds (0.1));
sourceApps.Stop (Seconds (49.8));
nrHelper->EnableTraces();
```

➔ Always Start the Sink Apps First and Stop the Source Apps First.

3.1.1.10 TCP Protocol

➔ Include tcp-rl.h and ns3/ns3-ai-module.h Header files

```
#include "tcp-rl.h"
#include "ns3/ns3-ai-module.h"
```

➔ Now set Default Configurations for TCP Protocol to be used

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType",
                TypeIdValue (TypeId::LookupByName (transport_prot)));
```

➔ Now TcpRlTimebased Congestion Control Algorithm will be used.

3.1.1.11 Flow Monitor

In NS-3, the Flow Monitor is a powerful tool for monitoring the network traffic at the flow level. It provides real-time or post-processing information about various network parameters such as throughput, packet loss, and delay.

The Flow Monitor works by capturing the packets passing through the network and analyzing them to extract flow-level statistics. A flow is defined as a set of packets that share a common set of characteristics, such as source and destination IP addresses, transport protocol, and port numbers.

To use the Flow Monitor in NS-3, we need to create an instance of the FlowMonitorHelper class, which helps to configure and install the monitor in the simulation. we can then use the FlowMonitorHelper object to install the monitor on a specific set of nodes or interfaces.

```
Ptr<FlowMonitor> flowmonitor;
```

```
FlowMonitorHelper flowmonHelper;
// To keep to track of flow from UE's to remoteHost and vice versa
NodeContainer endPoints;
endPoints.Add(remoteHost);
endPoints.Add(ueNodes);
flowmonitor = flowmonHelper.Install (endPoints);
```

Once the Flow Monitor is installed, we can start the simulation and collect data. The collected data can be used to generate various types of reports, including a flow-level report that provides information about each flow in the network. The report can be saved to a file.

3.1.1.12 Simulation

➔ Now Start the Simulation

```
Simulator::Stop (Seconds (simTime));
Simulator::Run ();
```

3.1.2. Python Setup

➔ For the most common use case, the ns-3 generates the data and then writes it to the shared memory, and the AI part reads the data from the shared memory to train the module.

➔ Structure of Data sent by ns-3 to shared memory

```
class TcpRlEnv(Structure):
  _pack_ = 1
  _fields_ = [
    ('nodeId', c_uint32),
    ('socketUid', c_uint32),
    ('envType', c_uint8),
    ('simTime_us', c_int64),
    ('ssThresh', c_uint32),
    ('cWnd', c_uint32),
    ('segmentSize', c_uint32),
    ('segmentsAcked', c_uint32),
    ('bytesInFlight', c_uint32),
  ]
```

➔ Structure of Data sent by python to shared memory

```
class TcpRlAct(Structure):
  _pack_ = 1
  _fields_ = [
    ('new_ssThresh', c_uint32),
    ('new_cWnd', c_uint32)
  ]
```

➔ This class establishes an environment for ns-3 and python to exchange data with the shared memory to implement Deep Learning algorithm with python

```
Init(1234, 4096)
var = Ns3AIRL(1234, TcpRlEnv, TcpRlAct)
```

- ➔ This class sets up an ns-3 environment and builds the shared memory pool.
- ➔ Runs ns-3 script in command prompt (terminal).
- ➔ param[in] shmKey : shared memory key
- ➔ param[in] memSize : shared memory Size
- ➔ param[in] programName : program name of ns3
- ➔ param[in] path : current working directory

```
exp = Experiment(1234, 4096, 'nr-tcp-rl-uplink', '../../')
exp.run(show_output=1)
```

3.2 BUILD AND RUN

- ➔ Build the Project
  - ◆ In ns-3-dev/ folder execute the following command ./waf to build the project
  - ◆
    ```
    Waf: Entering directory `/home/*/ns-3-dev/build'
    [2203/2258] Compiling scratch/nr-tcp-rl/sim.cc
    [2216/2258] Linking build/scratch/nr-tcp-rl/nr-tcp-rl
    Waf: Leaving directory `/home/groot/ns-3-dev/build'
    Build commands will be stored in build/compile_commands.json
    'build' finished successfully (7.563s)
    ```

- ➔ Run the Project
  - ◆ In ns-3-dev/scratch/{nr-tcp-rl}/ folder execute the following command
  - ◆ `python3 run_rl_tcp.py --use_rl --result`
    - arg --use_rl : To enable RL algorithm
    - arg --result : To generate and store results

4. RESULT DISCUSSION

To Compare the performance of our TcpRlTimeBased protocol with other protocols we have considered the following metrics:

1. Average BSR per packet
2. End-to-End Delay
3. Packet Delivery Ratio
4. Percentage of SR

Along with these we have plotted cWnd and instantaneous throughput with respect to time. We have simulated the entire scenario for 10, 20 and 30 UE’s and numerologies 1 and 2.

4.1 End to End Delay

End-to-end delay refers to the total time it takes for a packet of data to travel from the source device to the destination device in a computer network. It includes the time required for the source device to prepare the packet for transmission, the time it takes for the packet to travel through the network (including any delays caused by network congestion or other factors), and the time it takes for the destination device to process the packet once it has been received.

End-to-end delay is typically measured in milliseconds (ms) or microseconds (µs) and can be affected by a number of factors, including network bandwidth, latency, and the processing speed of the devices at either end of the network. In some applications, such as real-time video or audio streaming, minimizing end-to-end delay is critical to ensuring a high-quality user experience.

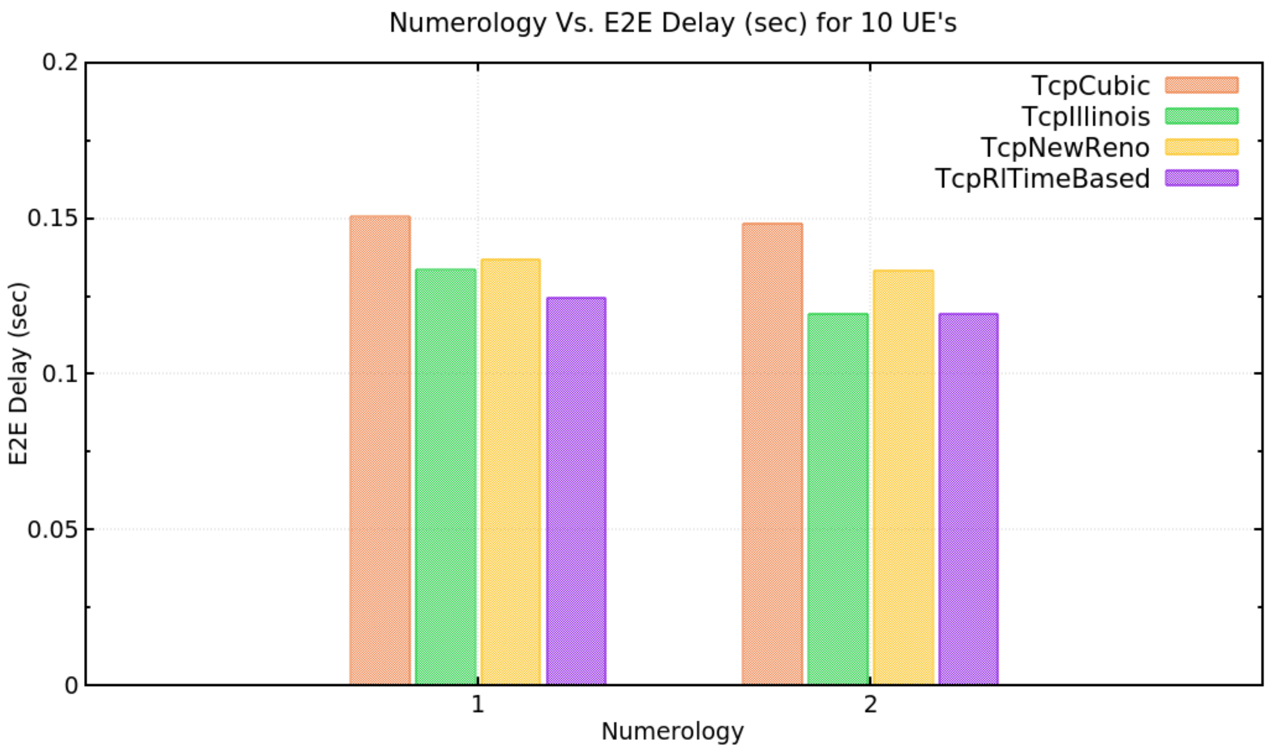

Fig 3. End-to-End Delay vs Numerology for 10 UE’s

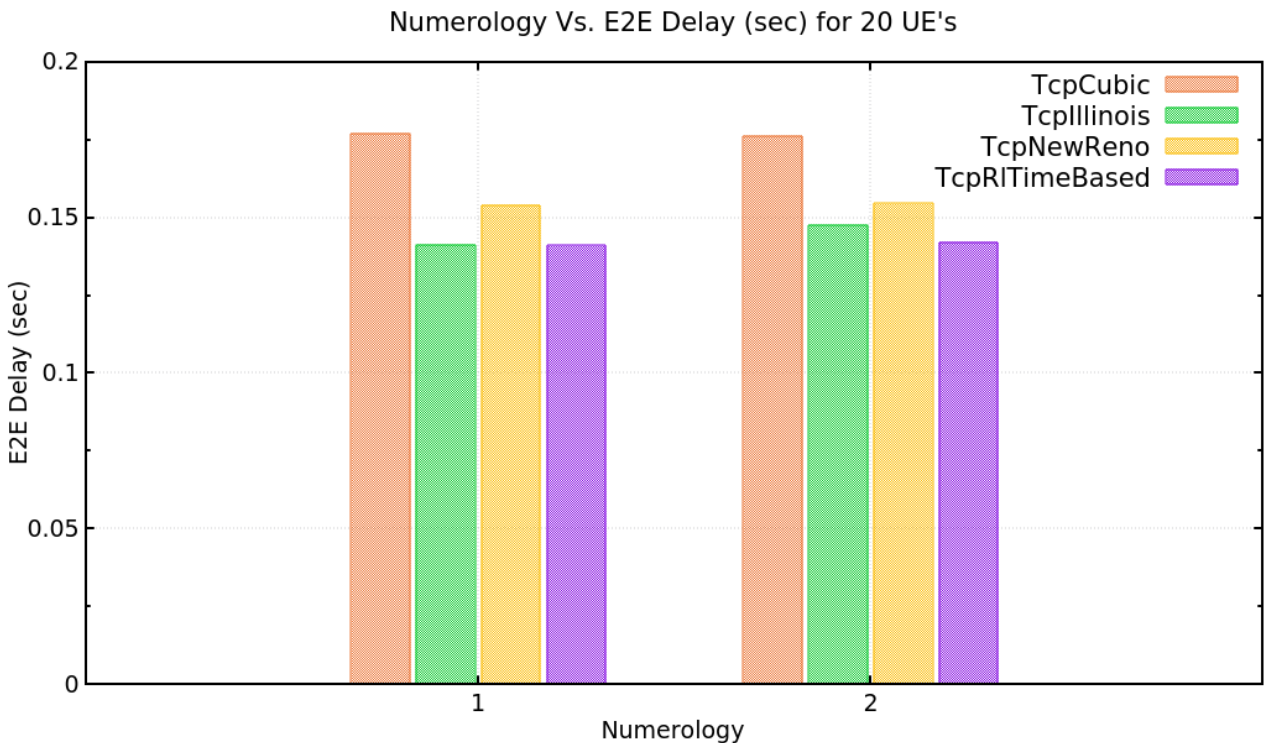

Fig 4. End-to-End Delay vs Numerology for 20 UE's

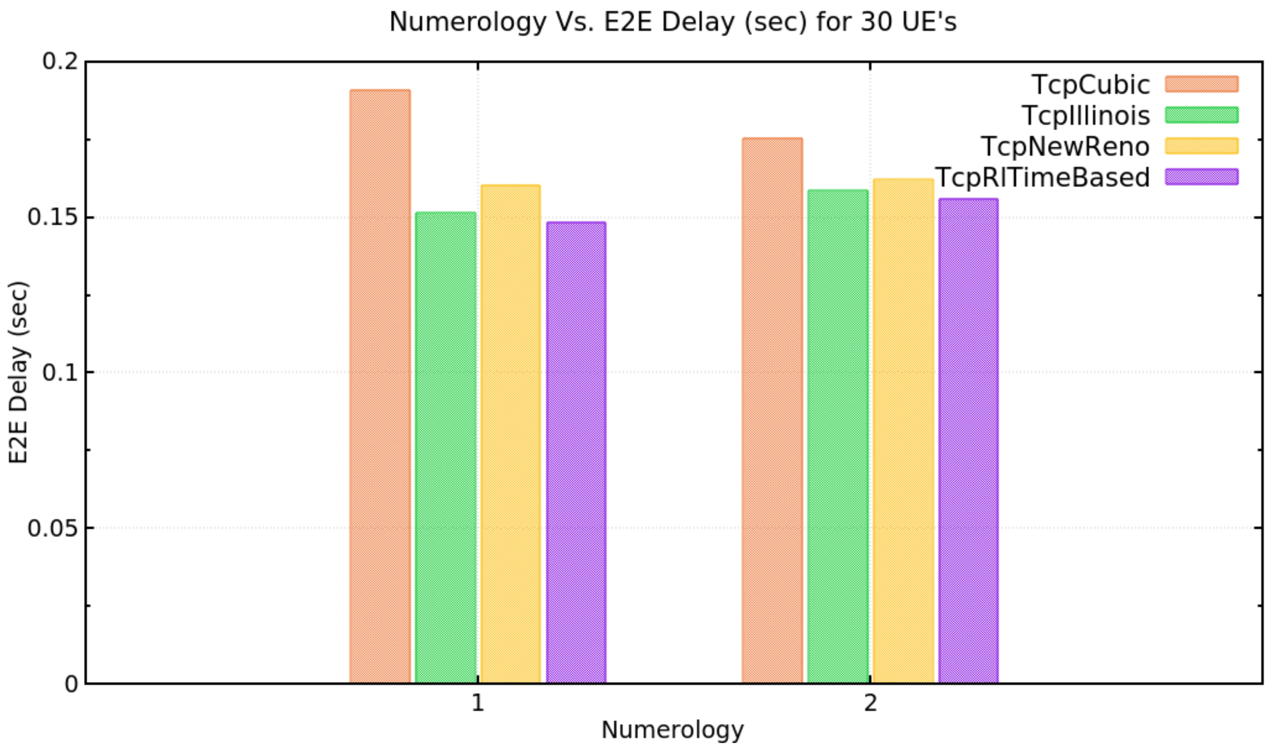

Fig 5. End-to-End Delay vs Numerology for 30 UE's

4.2 Percentage of Scheduling Requests (SR)

In mobile communication networks, a Scheduling Request (SR) is a message sent by a User Equipment (UE) to request an allocation of radio resources from the base station (gNodeB) for uplink transmission. The SR is used in radio resource management to ensure efficient allocation of resources to the UE.

When a UE has data to transmit, it sends a scheduling request to the base station, indicating that it requires radio resources to transmit the data. The base station then responds with an allocation of radio resources, allowing the UE to transmit its data. The scheduling request may include information such as the type of data to be transmitted, the quality of service required, and the amount of data to be transmitted.

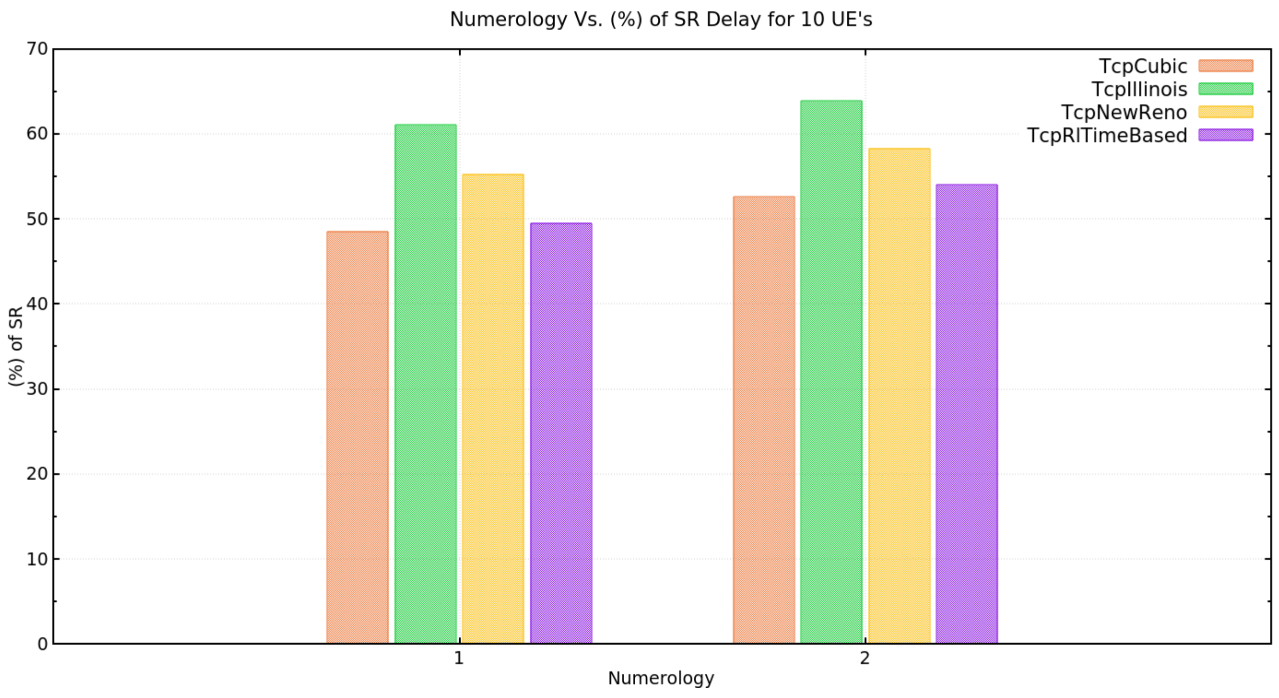

Fig 6. Percentage of SR vs Numerology for 10 UE’s

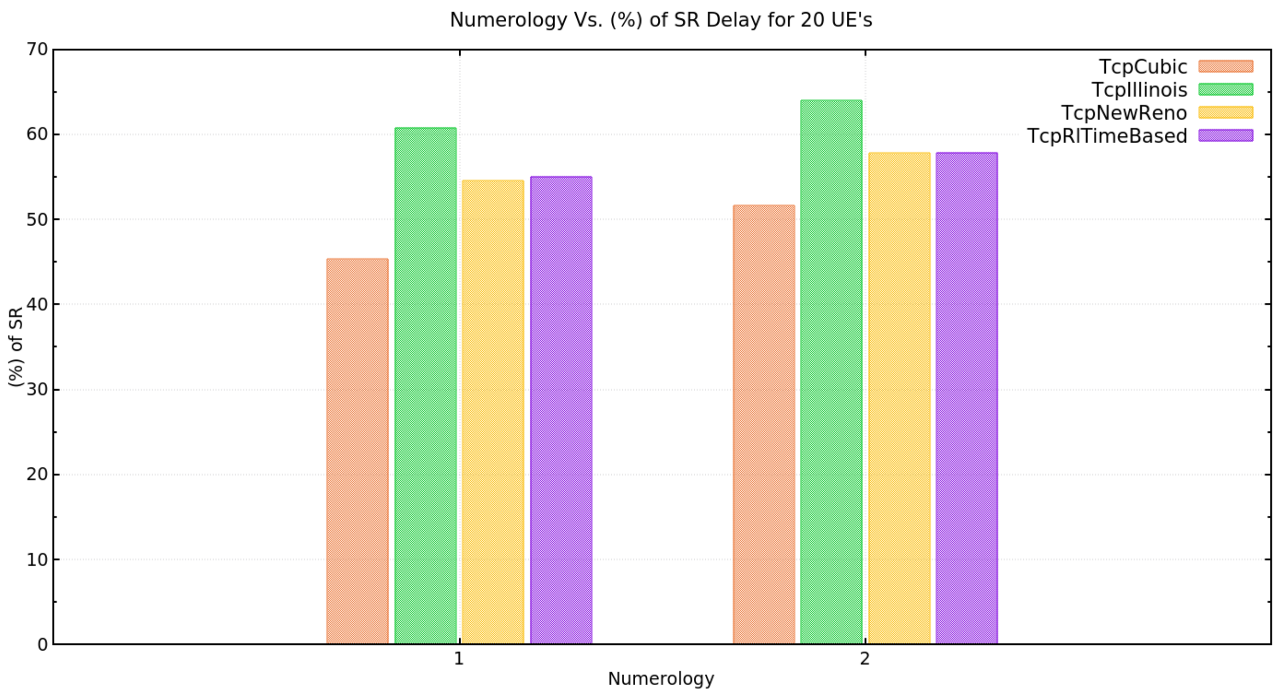

Fig 7. Percentage of SR vs Numerology for 20 UE's

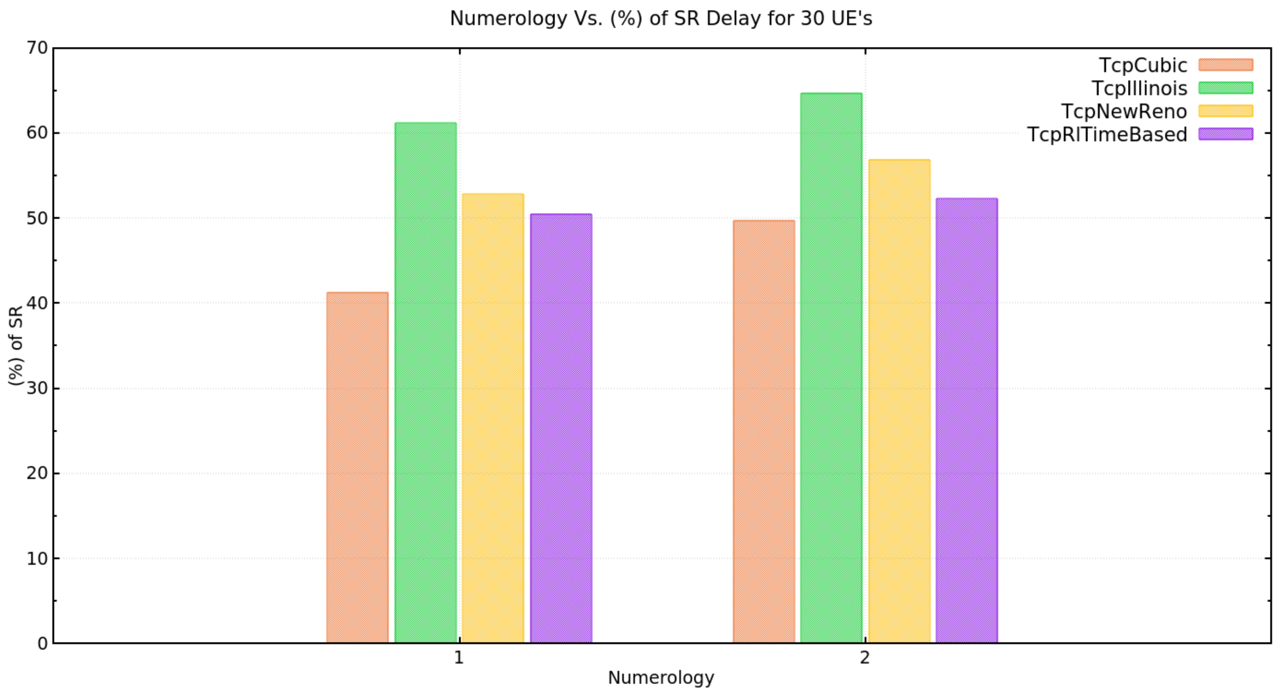

Fig 8. Percentage of SR vs Numerology for 30 UE's

4.3 Average BSR per Packet

BSR (Buffer Status Reporting) is a mechanism used in the mobile communication networks to provide the base station (gNodeB) with information about the amount of data buffered in the user equipment (UE).

The UE is required to periodically report its buffer status to the base station using BSR messages. The BSR message contains information about the amount of data buffered in the UE, such as the number of bytes or the number of packets waiting to be transmitted.

The base station uses this information to allocate resources for the UE and to prioritize the transmission of data.

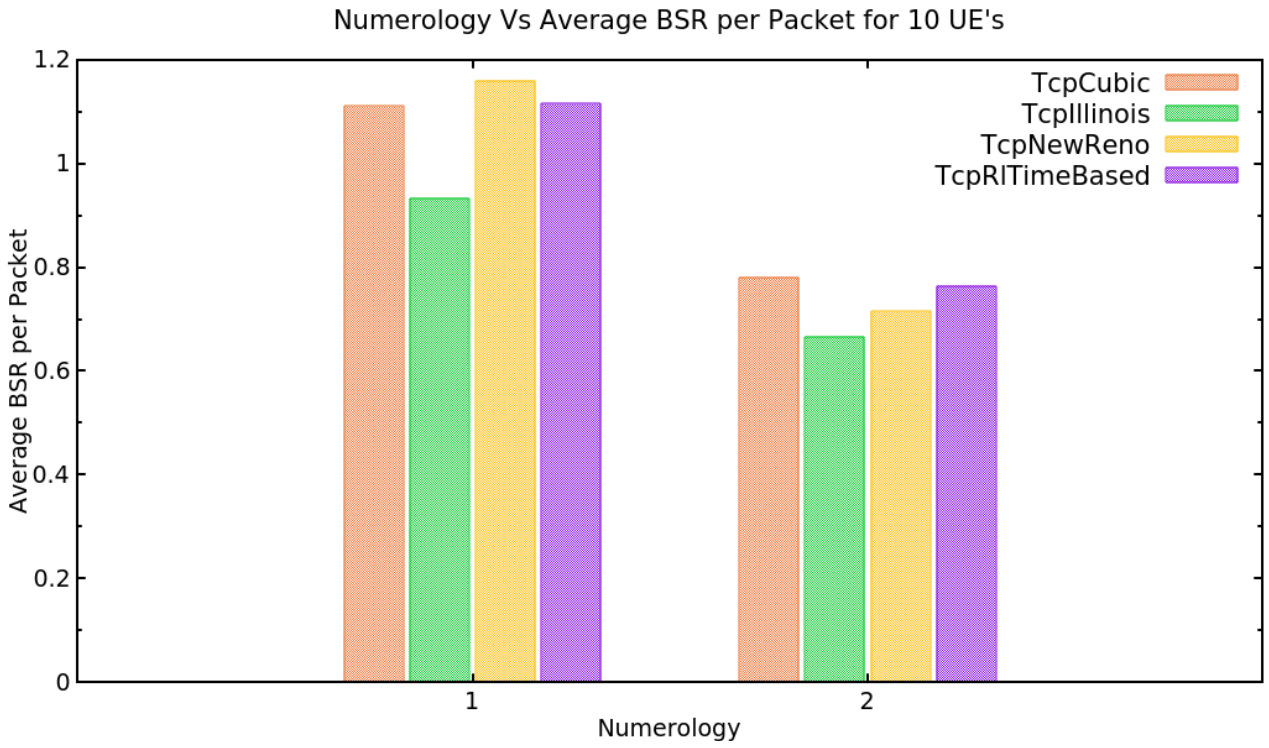

Fig 9. Average BSR per Packet vs Numerology for 10 UE's

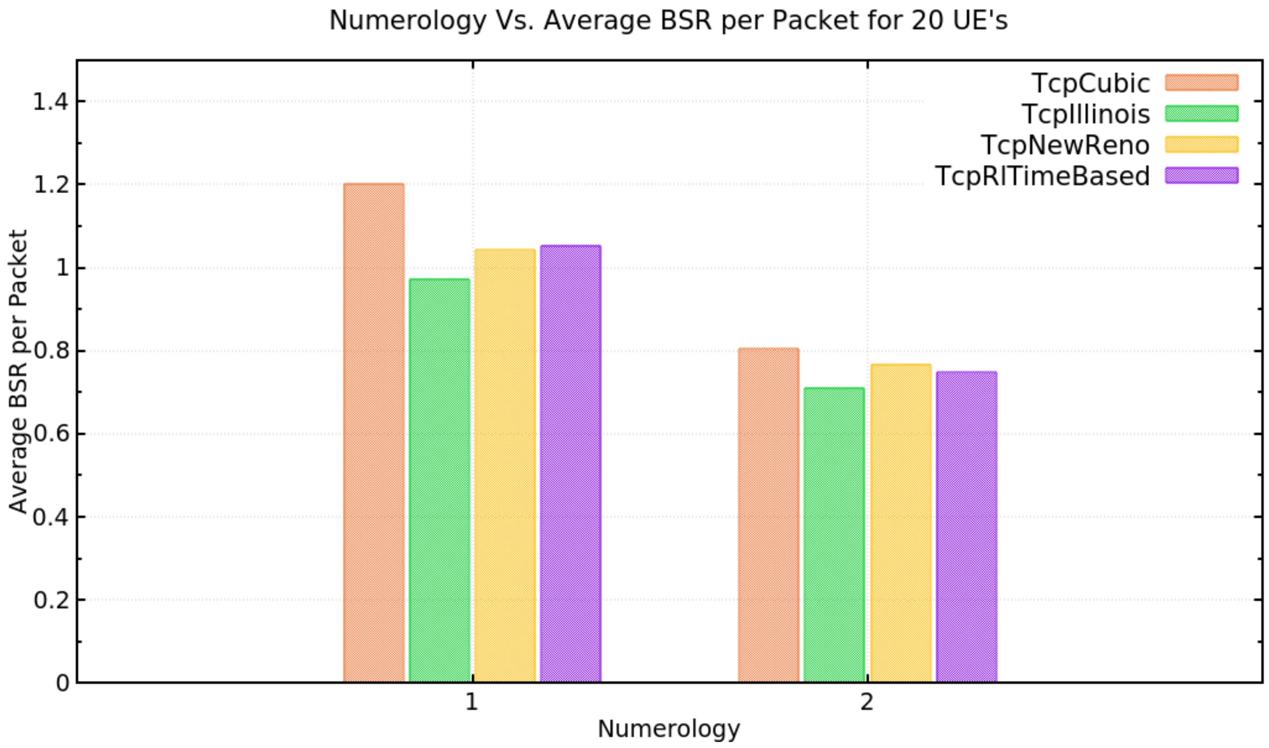

Fig 10. Average BSR per Packet vs Numerology for 20 UE's

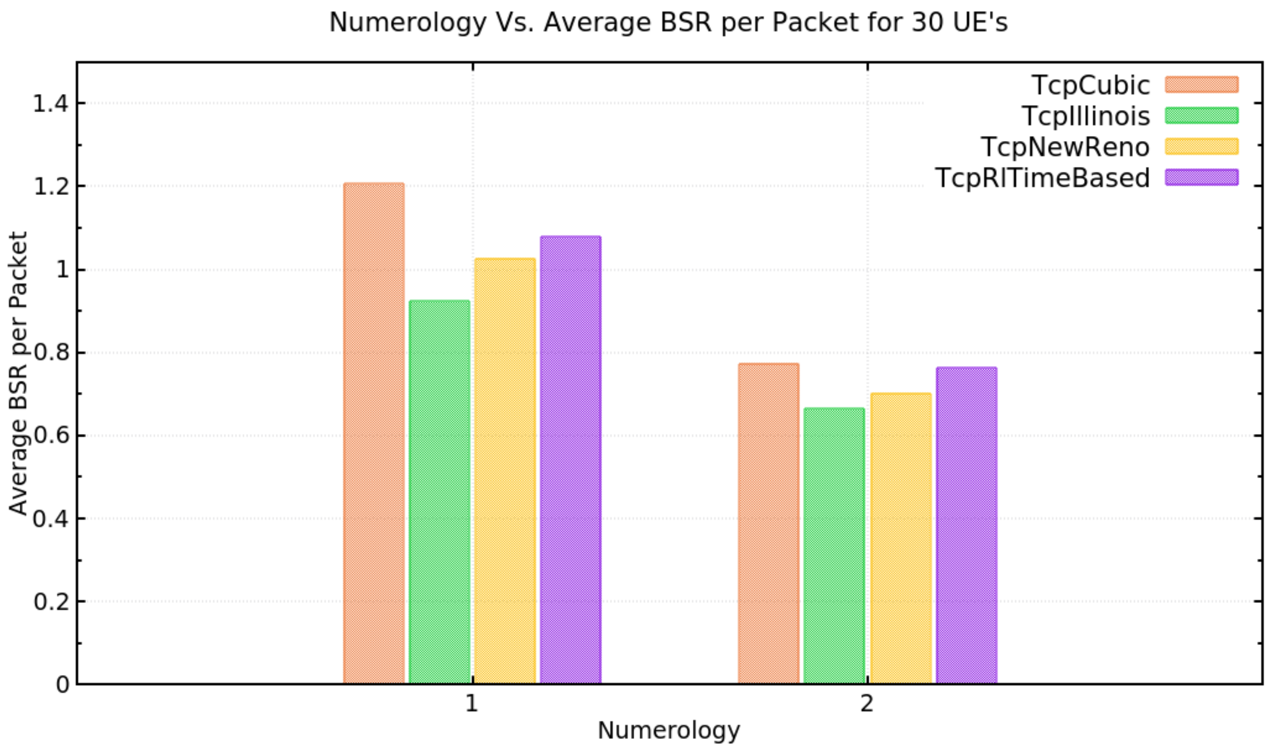

Fig 11. Average BSR per Packet vs Numerology for 30 UE's

4.4 Instantaneous Throughput

Instantaneous throughput is a measure of the rate at which data is being transmitted over a communication channel at a particular point in time. It is typically measured in bits per second (bps), and it reflects the current data transfer rate between the sender and receiver.

The instantaneous throughput can vary depending on a number of factors, including the available bandwidth, network congestion, packet loss, and other factors that may affect the quality of the communication channel.

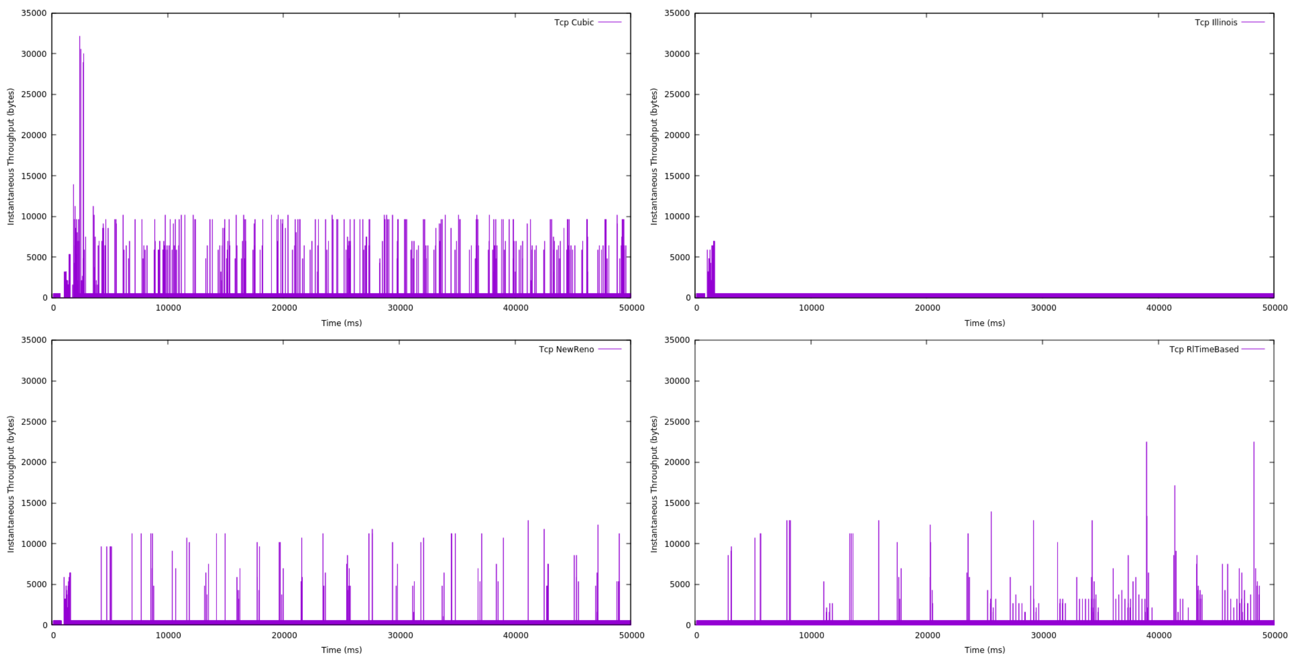

Fig 12. Instantaneous Throughput vs Time for 10 UE's and Numerology 1

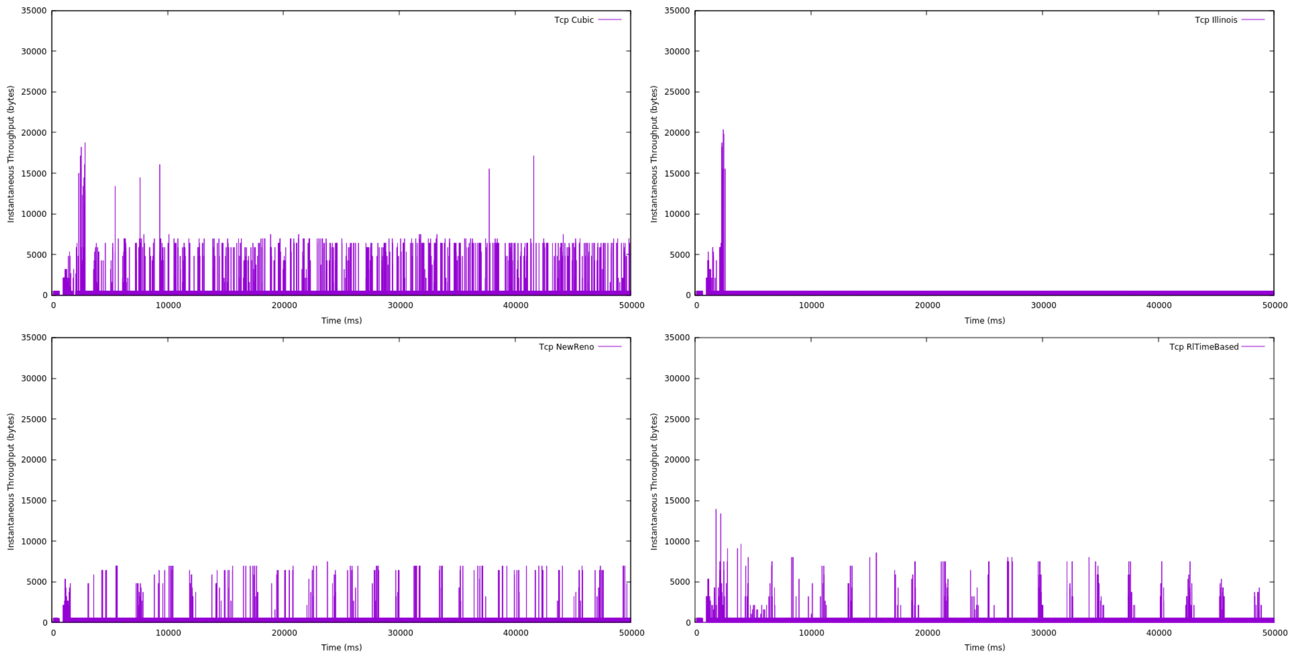

Fig 13. Instantaneous Throughput vs Time for 20 UE's and Numerology 1

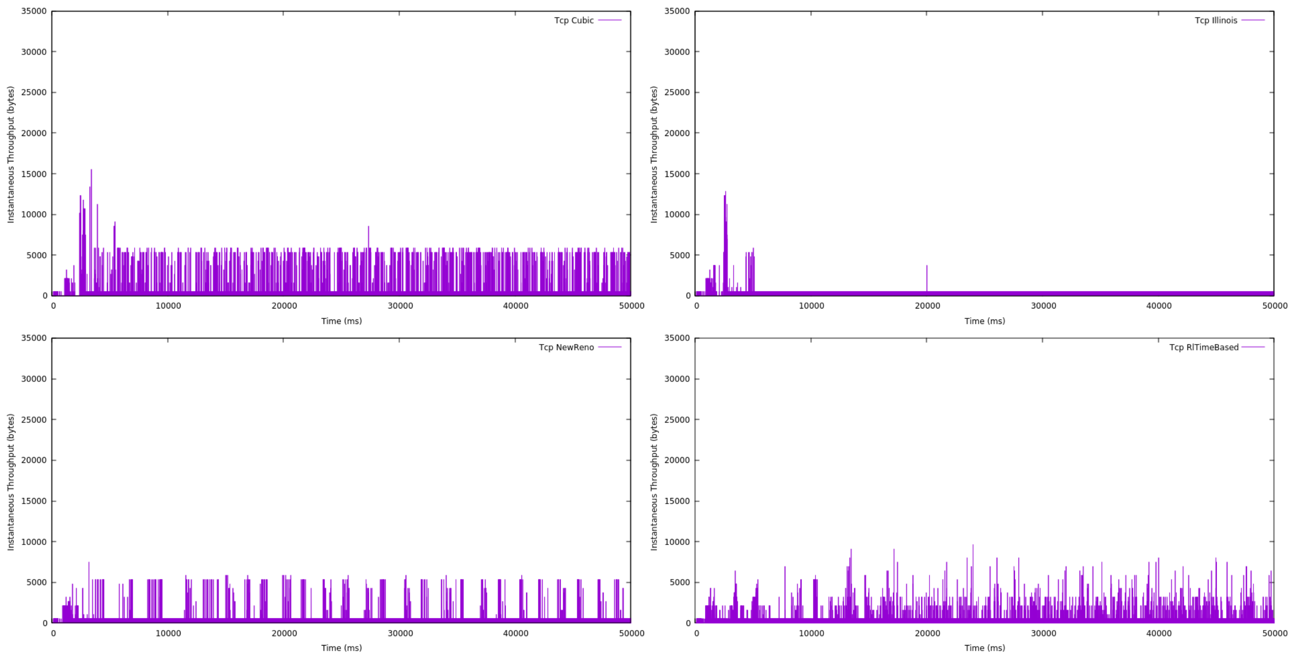

Fig 14. Instantaneous Throughput vs Time for 30 UE's and Numerology 1

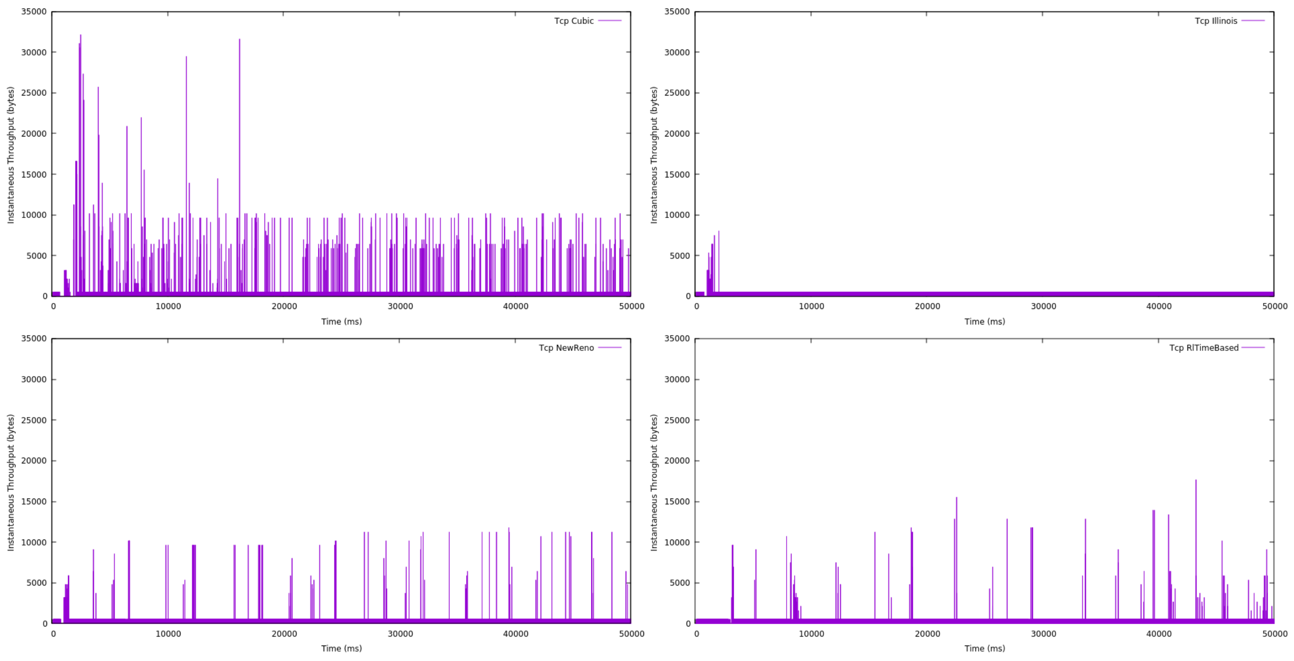

Fig 15. Instantaneous Throughput vs Time for 10 UE’s and Numerology 2

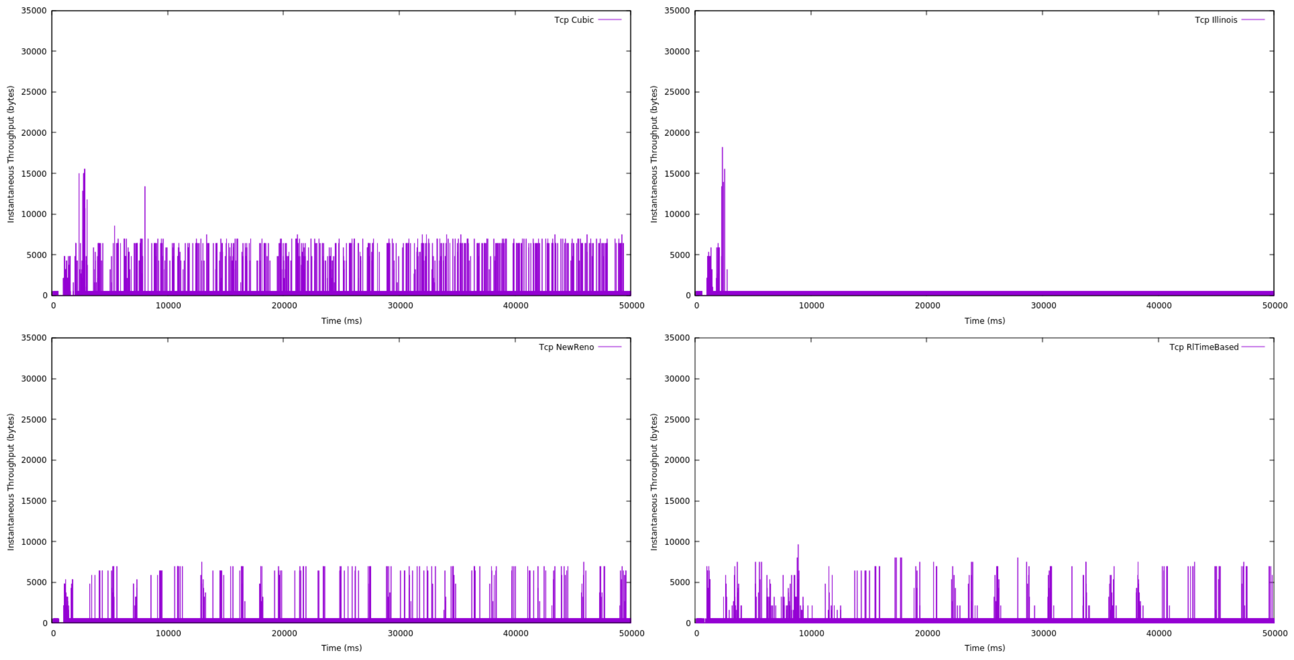

Fig 16. Instantaneous Throughput vs Time for 20 UE’s and Numerology 2

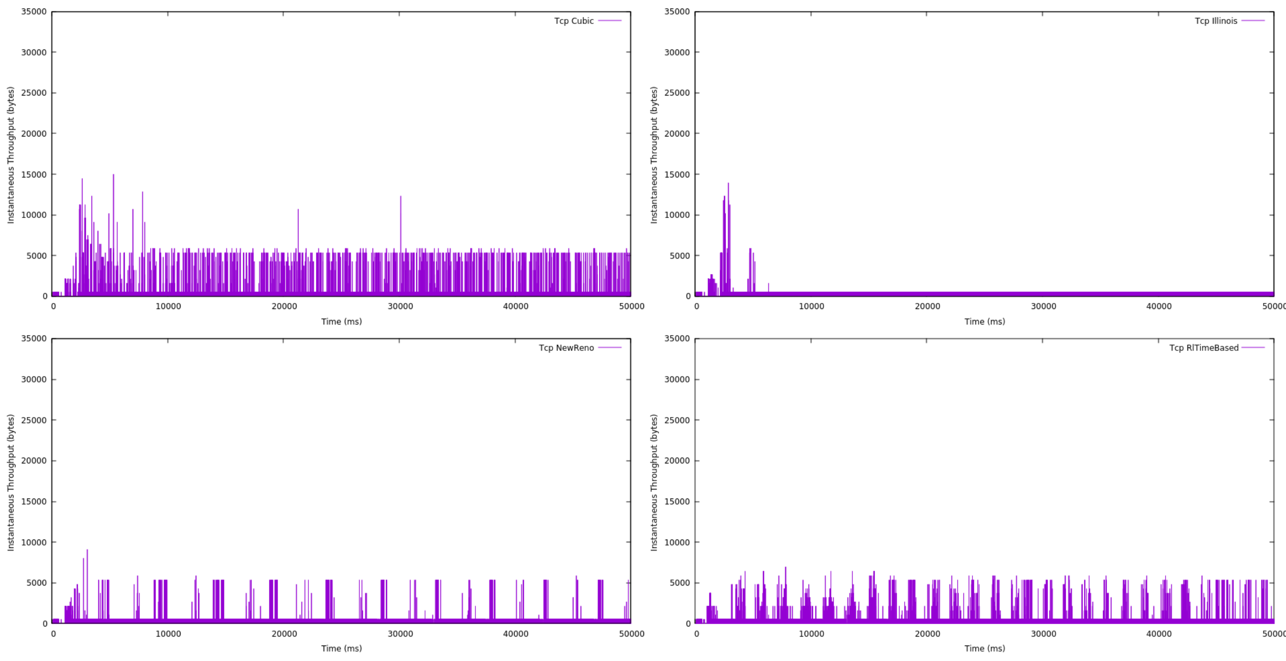

Fig 17. Instantaneous Throughput vs Time for 30 UE's and Numerology 2

4.5 Congestion Window

The congestion window is a mechanism used in the TCP transport layer to control the rate at which data is transmitted over a network connection. It helps prevent network congestion and ensures fair resource allocation by dynamically adjusting the number of packets that can be in transit at any given time based on current network conditions.

The congestion window size is dynamic and is adjusted based on the current network conditions. If the network is congested, the window size is reduced, and if the network is not congested, the window size is increased. This allows the sender to efficiently utilize the available network bandwidth while also preventing congestion and ensuring fair resource allocation among competing network flows.

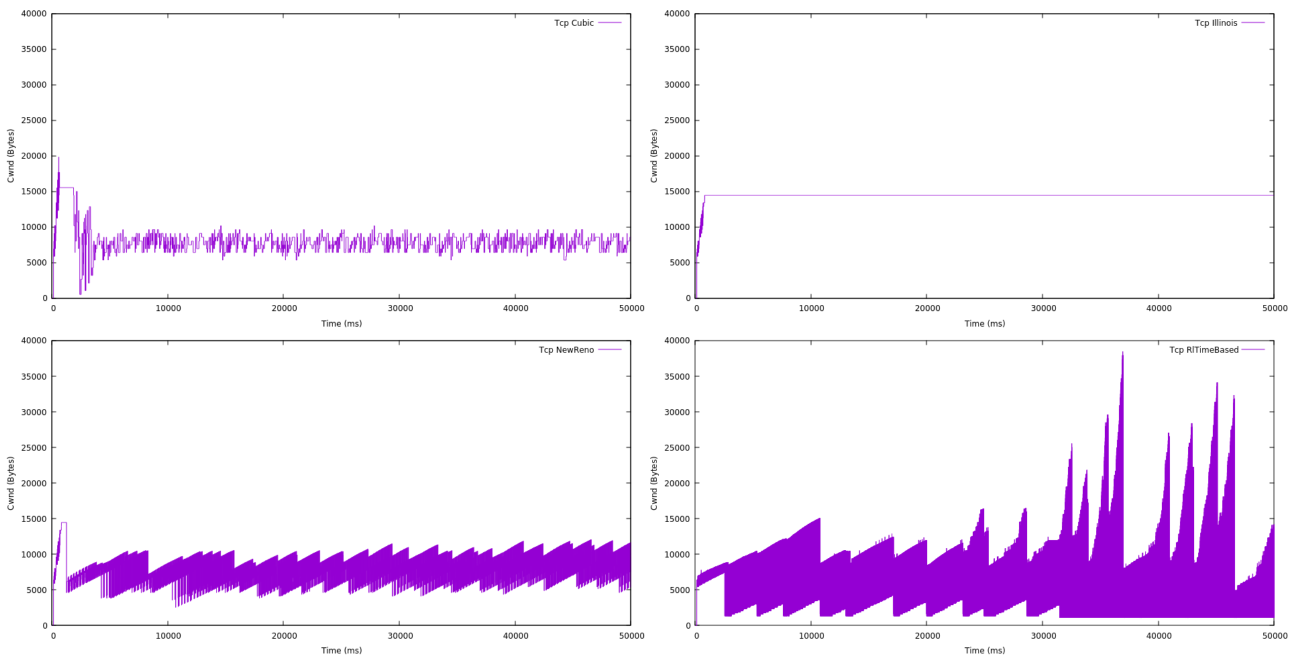

Fig 18. Cwnd vs Time for 10 UE’s and Numerology 1

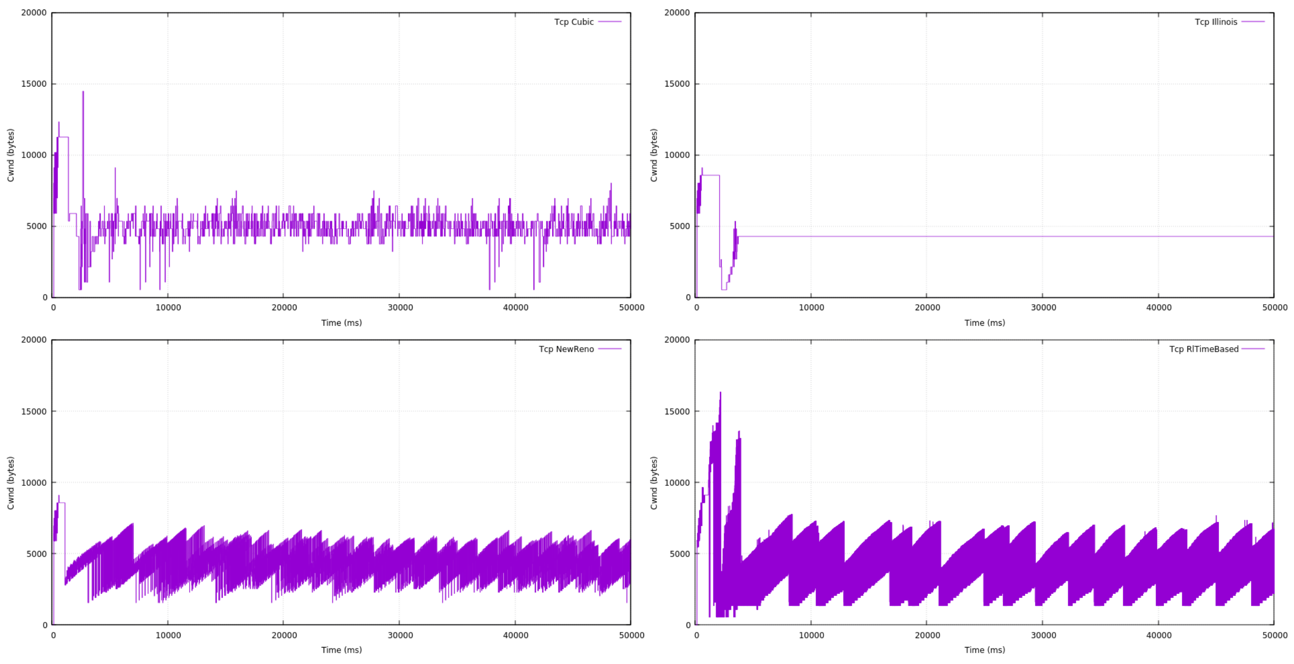

Fig 19. Cwnd vs Time for 20 UE’s and Numerology 1

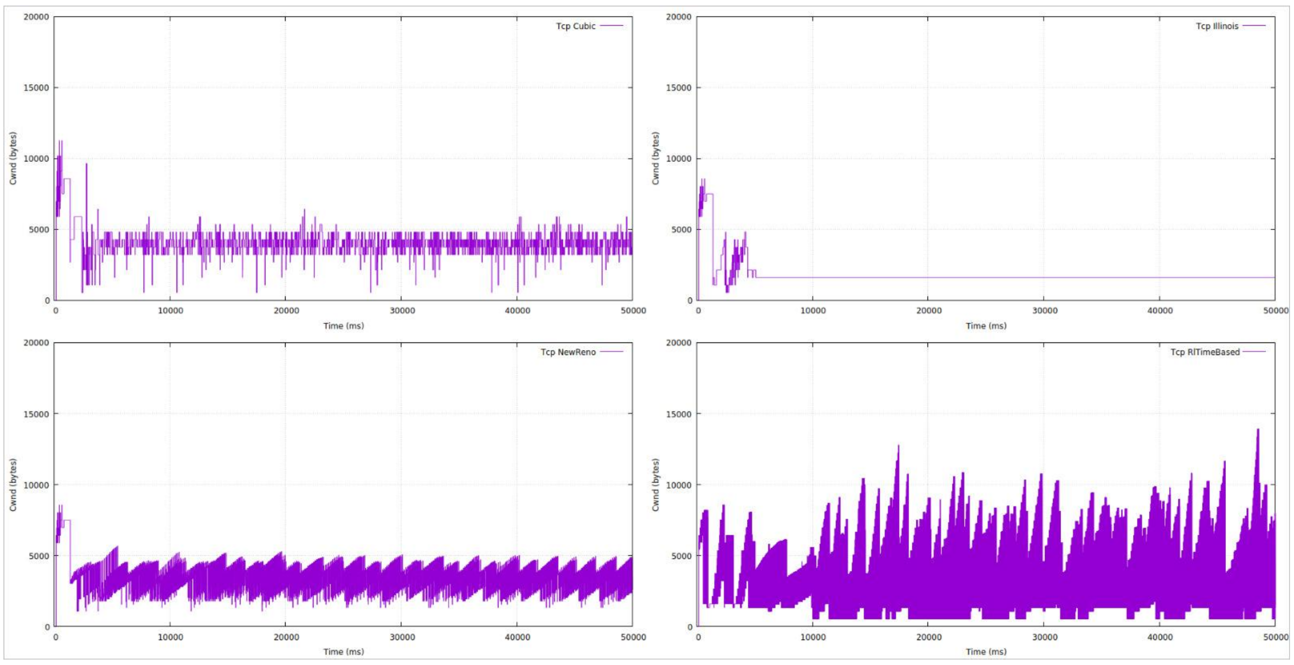

Fig 20. Cwnd vs Time for 30 UE's and Numerology 1

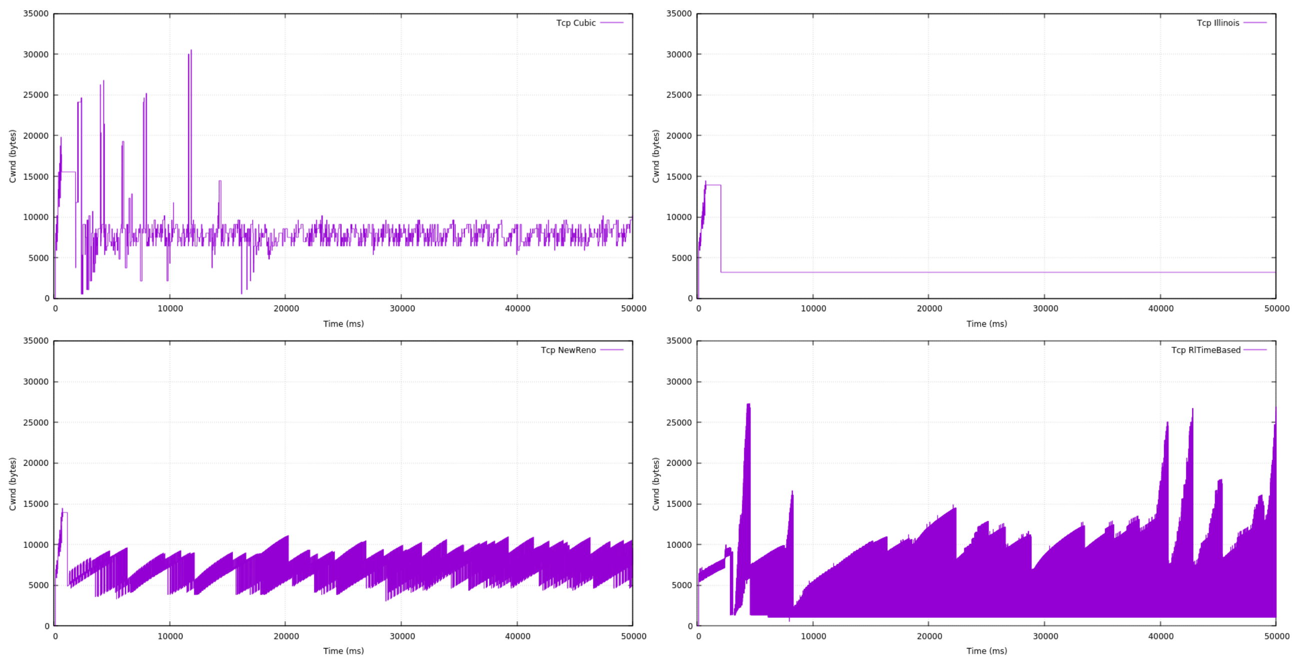

Fig 21. Cwnd vs Time for 10 UE's and Numerology 2

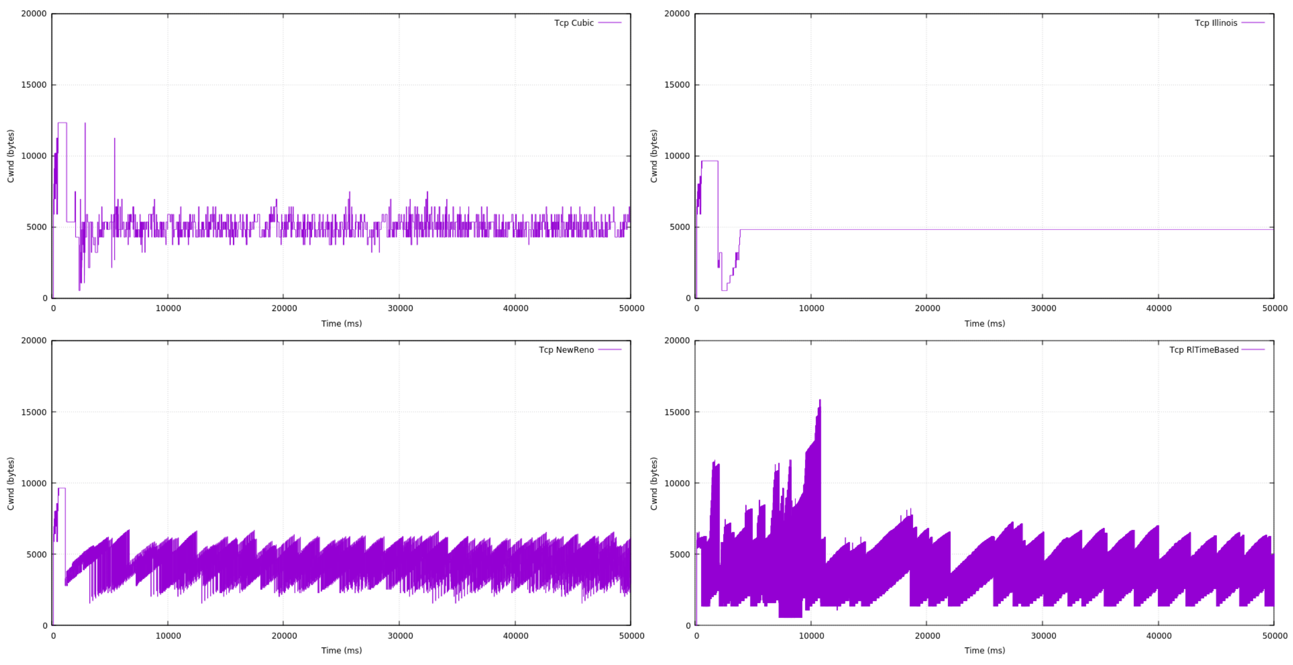

Fig 22. Cwnd vs Time for 20 UE's and Numerology 2

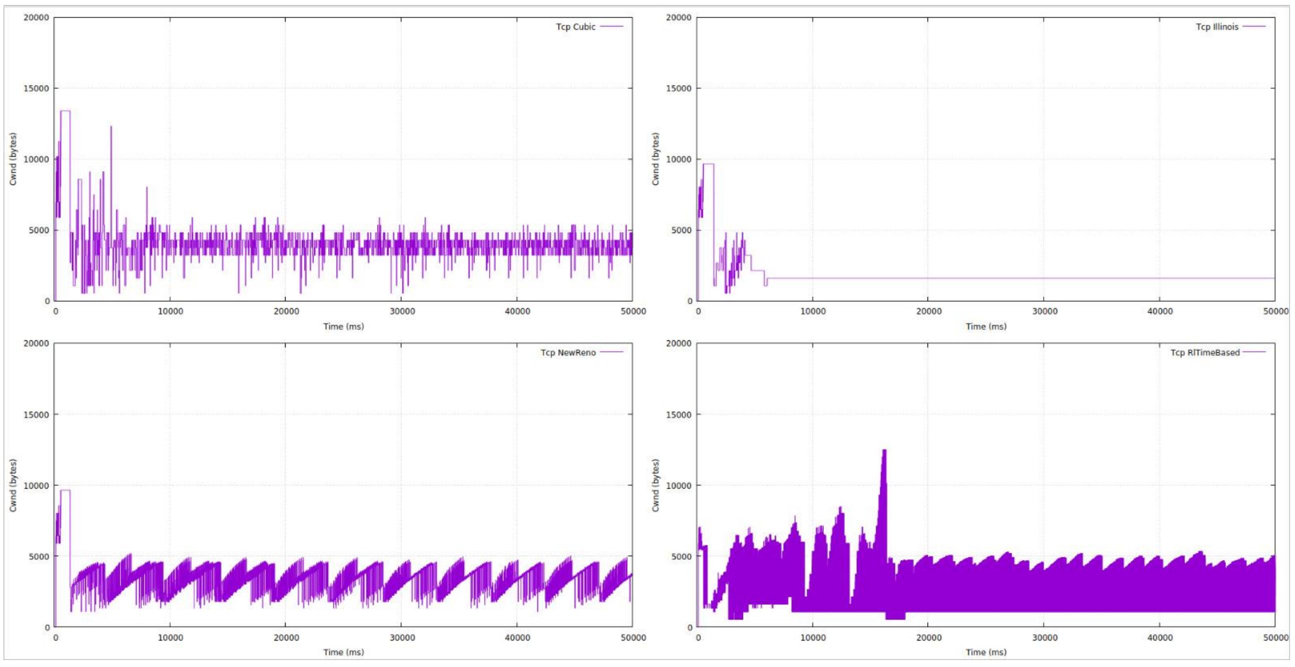

Fig 23. Cwnd vs Time for 30 UE's and Numerology 2

4.6 Packet Delivery Ratio

Packet Delivery Ratio (PDR) is a metric used to measure the percentage of packets that are successfully delivered to their intended destination in a network. It is calculated by dividing the number of packets that are successfully delivered by the total number of packets that were sent.

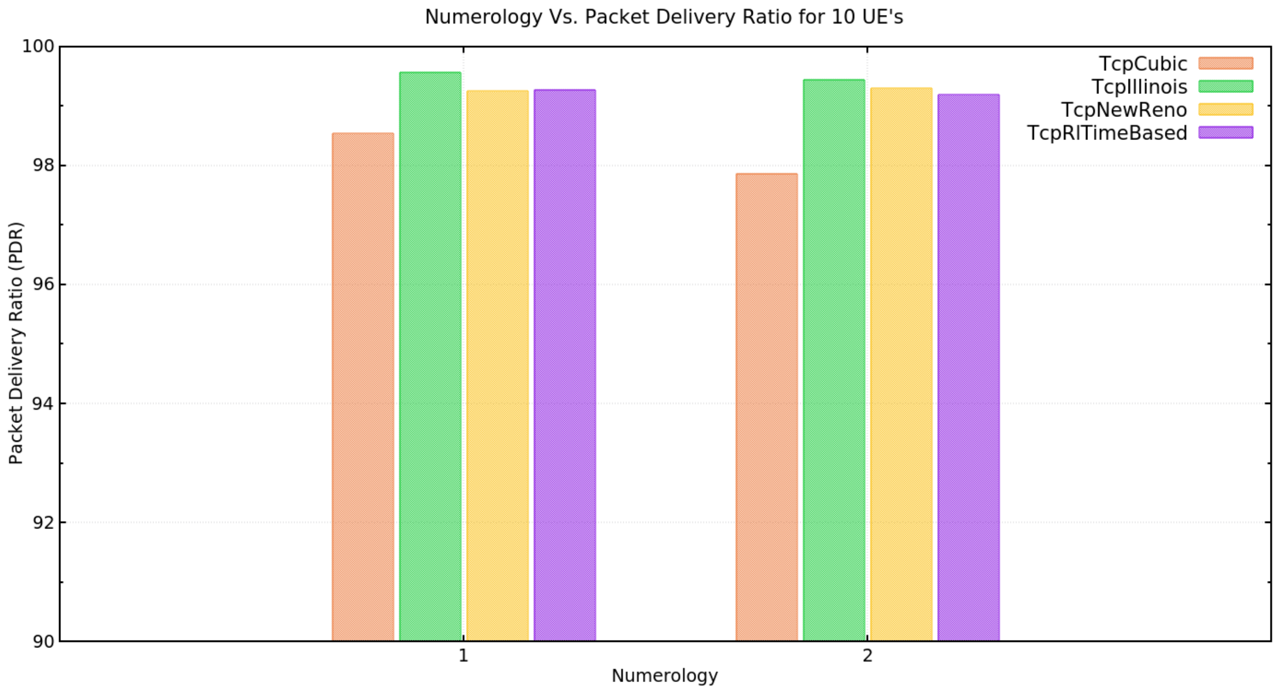

Fig 24. Packet Delivery Ratio vs Numerology for 10 UE’s

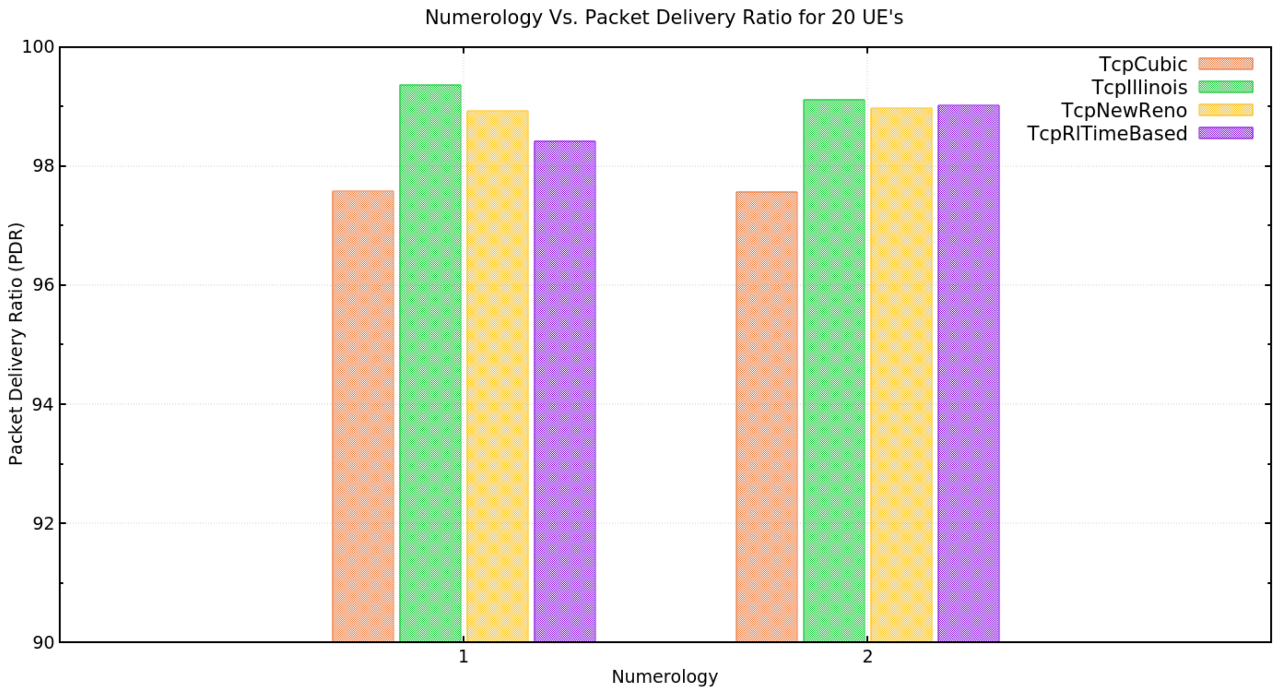

Fig 25. Packet Delivery Ratio vs Numerology for 20 UE’s

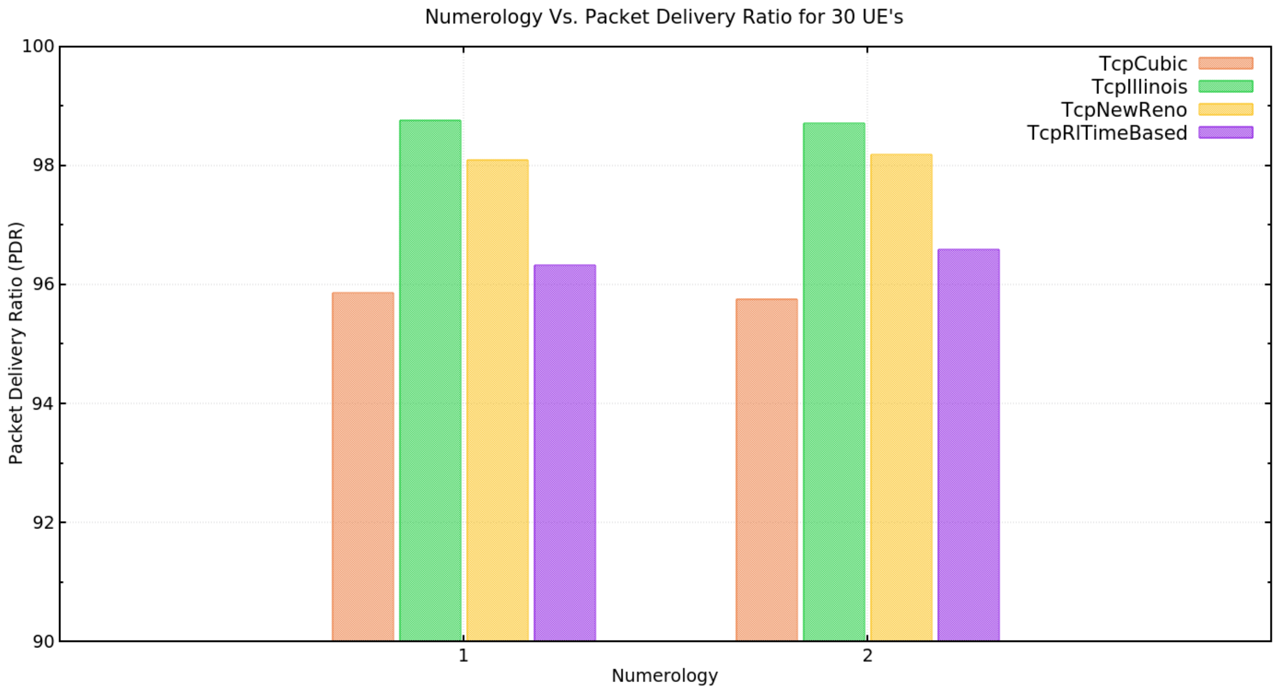

Fig 26. Packet Delivery Ratio vs Numerology for 30 UE’s

5. CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this project we have taken the RlTimeBased algorithm and placed it against the other tcp algorithms to test its strengths over the other algorithms. The RlTimeBased has outperformed other protocols in case of end-to-end delay and gave a tough competition to other protocols in the other metrics such as percentage of SR, average BSR per packet, packet delivery ratio, and RLC-to-RLC delay. More importantly the RlTimeBased algorithm almost beat the other algorithms with no exposure to real time data and has the potential to beat all the algorithms if trained using real time data.

There is a significant growth in Average BSR per packet in numerology 2 as compared to numerology 1. Thus, lower numerologies are more suited for applications with low mobility, such as fixed wireless access, while higher numerologies are better suited for high mobility applications, such as connected vehicles.

The RLTimeBased has a better packet delivery ratio in numerology 2 as compared to numerology 1 while the other protocols remain approximately the same. We can conclude that the numerology determines the subcarrier spacing and the duration of the TTI. Higher numerologies have shorter TTIs, which means that data can be transmitted more frequently, reducing the delay between transmissions and improving the overall quality of service for mobile users.In addition, higher numerologies also allow for higher bandwidths, which can improve the data transfer rate and enable the transmission of more data per unit of time.

RLTimeBased has efficiently handled the congestion window and changes the size of the congestion window according to traffic in the simulation which has resulted in maximum throughput as compared to other protocols.

5.2 Future Scope

Machine learning (ML) has the potential to revolutionize computer networks in a number of ways. Here are some potential future scopes of ML in computer networks:

1. Quality of service (QoS): ML algorithms can be used to optimize QoS by predicting network congestion and dynamically adjusting bandwidth allocation to ensure that high-priority traffic receives the necessary bandwidth.
2. Traffic engineering: ML algorithms can be used to analyze network traffic patterns and optimize network routing, load balancing, and congestion control algorithms.
3. Network analytics: ML algorithms can be used to analyze network performance data and identify patterns, anomalies, and trends that can be used to improve network performance and troubleshoot issues.

REFERENCES

[1] Hao Yin, Lyutian yang Zhang, Xiaojun Hei, Yayu Gao, “ns3-ai: Fostering Artificial Intelligence Algorithms for Network Research”, DOI:10.1145/3389400.3389404

[2] Martin Sauter, “FROM GSM TO LTE: An Introduction to Mobile Networks and Mobile Broadband”, John Wiley and Sons, Ltd, 2011.

[3] Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction”, MIT Press, 2014, 2015.

[4] https://www.geeksforgeeks.org/what-is-tcp-new-reno/

[5] https://www.cs.princeton.edu/courses/archive/fall16/cos561/papers/Cubic08.pdf

[6] https://www.nsnam.org/docs/models/html/tcp.html

[7] https://apps.nsnam.org/app/ns3-ai/

[8] https://cttc-lena.gitlab.io/nr/html/classns3_1_1_nr_helper.html

[9] http://www.gnuplot.info/

[10] https://www.nsnam.org/docs/models/html/flow-monitor.html

[11] https://www.nsnam.org/docs/release/3.19/doxygen/classns3_1_1_bulk_send_helper.html

[12] https://www.nsnam.org/docs/release/3.18/doxygen/classns3_1_1_packet_sink.html

[13] https://www.nsnam.org/docs/release/3.18/doxygen/classns3_1_1_application_container.html

[14] https://www.nsnam.org/docs/release/3.16/doxygen/classns3_1_1_random_walk2d_mobility_model.html

[15] https://www.nsnam.org/docs/release/3.16/doxygen/classns3_1_1_constant_position_mobility_model.html