UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Analysis of TCP Performance for LTE-5G Millimeter Wave Dual Connectivity

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Merly Asunción Rojas

In partial fulfilment

of the requirements for the degree of

MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor: Ilker Demirkol

Barcelona, October 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

telecom BCN

Title of the thesis: Analysis of TCP performance for LTE-5G Millimeter Wave Dual Connectivity

Author: Merly Marilyn Asuncion Rojas

Advisor: Ilker Demirkol

Abstract

The goal of this work is the analysis of the performance of the transport control protocol (TCP) in a Dual connectivity (DC) system, where both LTE and 5G millimeter wave (mmWave) were used in the radio access network, while a single user travels across the scenario. Since the user is moving, the interaction between the mmWave base stations (BSs) must be very efficient to avoid congestion events. This makes the analysis of DC very important.

Simulation models based on open-source software frameworks were used to evaluate the performance of Dual connectivity for a 5G non-standalone (NSA) solution, where all the 5G base station traffic goes through the LTE base station. The scenarios proposed were defined in terms of non-line-of-sight/line-of-sight (NLOS/LOS) scenario, medium/high traffic, which are used to evaluate different TCP congestion control algorithms. The performance was then evaluated in terms of goodput, packet delivery ratio, standard deviation of bytes in-flight, and round-trip time. Simulation results showed that the number of bytes in-flight grows with high rates and large latencies caused by inter-BS communication. The mmWave medium is very sensitive to channel conditions specially in the middle point between mmWave BSs causing ping-pong effect during a handover (HO). At the beginning of the simulation some nodes overflow due to the aggressive slow start mechanisms, which turn to be very problematic for high traffic rates. In that sense, TCP Cubic proves to be a much reliable congestion control algorithm since it implements a hybrid slow start method.

<u>Acknowledgements</u>

I would like to express my sincere thanks to my advisor, Prof Ilker Demirkol for his valuable support, guidance and patience throughout my thesis work. Also, thanks to my colleagues at the laboratory for their cooperation and assistance.

Finally, my parents and my lovely husband that always believe in me and support me throughout all the decisions I take.

Revision history and approval record

Revision	Date	Purpose
0	13/10/2019	Document creation
15	17/10/2019	Document revision

Written by: Merly Marilyn Asuncion Rojas		Reviewed and approved by: Ilker Demirkol	
Date	14/10/2019	Date	17/10/2019
Name	Merly Marilyn Asuncion Rojas	Name	Ilker Demirkol
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

In the first quarter of 2019, mobile data traffic represents approximately half of web traffic worldwide and in the second quarter Ericsson reported that the total number of mobile subscriptions exceeded 7.9 billion, with a net addition of 60 million subscriptions during the quarter.

The report also shows that the Mobile data traffic grew by 78 percent between the second quarter of 2018 and the second quarter of 2019, as we can see in Figure 1.1.1. The rising number of smartphone subscriptions and the increment of the average data volume per subscription drives the traffic growth, another element that contributes is the video content. The graph below shows total global monthly data and voice traffic from the last quarter of 2013 to second quarter of 2019, along with the year-on-year percentage change for mobile data.

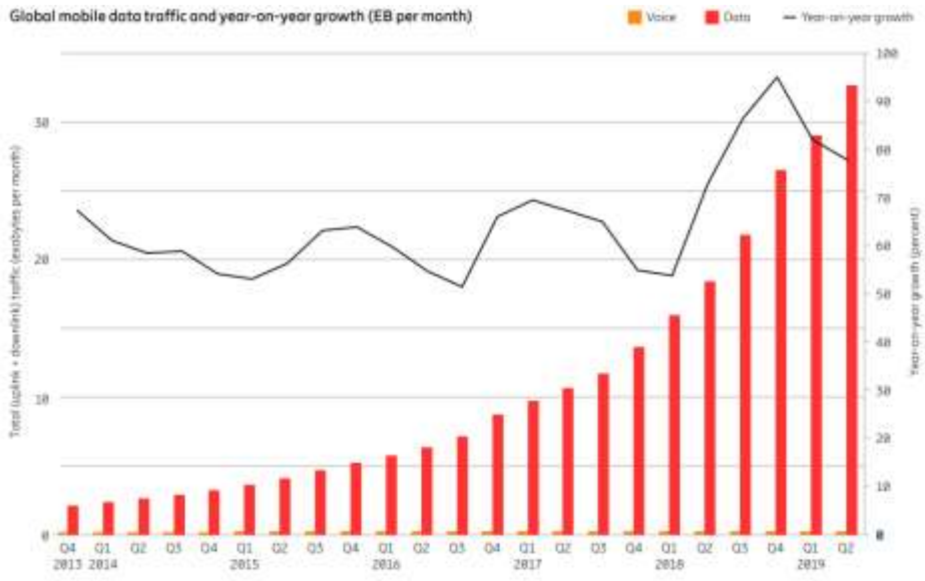


Figure 1.1.1: Global mobile data and voice traffic growth. (Source: The Ericsson Mobility Report, June 2019).

Due to the increment of mobile devices such as smartphones, tables, smartwatches, etc., the traffic demand is predicted to increase exponentially. Shannon's law says that the way to increment the capacity depends on the increment of the spectrum, enhancement of the spectral efficiency and reducing of the coverage per base station. Nowadays, the firsts two had almost reached the Shannon's limits, but the coverage area can be reduced by incrementing the small cell densification. The small cells deployment will depend on the radio frequency, small cell type and the internode connectivity architecture. 3GPP in Release 12 introduced "Dual connectivity" as a solution for the high per-user throughput demand.

The demand for more data, along with the spectrum scarcity, motivates the exploitation of new bands. The Millimetre wave (mmWave) bands offer the possibility of orders of magnitude greater throughput for next cellular systems. However, since mmWave signals are highly susceptible to blockage, channel quality on each mmWave link can be extremely intermittent. In order to guarantee reliable transmission for very high rates, the study of congestion algorithm becomes very important.

The transport control protocol (TCP) has many congestion control (CC) algorithms. All these variants basically differ in the way they deal with congestion, handle data rate, and how they deal with duplicate acknowledgment. Moreover, the distinction between TCP congestion states where the packet loss is due to congestion or corruption is also an issue to identify.

The ns-3 network simulator currently implements a wide range of network protocols across various layers of the communication network. Due to this it is a valuable tool for researchers working on cross-layer design. The network simulator 3 (ns-3) already hosts modules for the simulation of WiFi, WiMAX and 3GPP-LTE networks. This work uses the implementation of the ns-3 mmWave module by the New York University (NYU) and addresses the TCP performance characteristics needed to tackle the irregularity of using 5G millimeter wave in mobile networks.

1.2. <u>Statement of purpose</u>

5G Dual connectivity offers the capacity to reach high bit rates thanks to the different solutions it implements such as beamforming management, cell search, massive MIMO, higher frequency bands (Millimetre waves), etc.

This work presents a detailed investigation of the behaviour of distinct TCP implementation under various network conditions in LTE/5G deployment with Dual connectivity using Millimetre waves in the access network. Also, TCP's capability for adapting to the rapid variability of mobile networks under different network loads during were studied.

There are numerous possible parameter combinations that could be experimented in order to test their impact on the performance, but the study limited the scope to the following parameters: X2 latency (in terms of system response for handover), RLC buffer in AM (effects of buffering) and non-line-of-sight/line-of-sight.

This work aims to contribute on identifying critical design issues of congestion control algorithms. Also identifying the limitation factors of the user experience, three main objectives have been proposed:

- Analysis of ns-3 module for Dual connectivity and mmWave characteristics.
- Analysis of diverse TCP congestion control methods employed by ns-3.
- Performance analysis through certain performance metrics, e.g. round-trip time, congestion window, throughput, Bytes in-flight, throughput PDCP latency and confidence level.

1.3. <u>Requirements and specifications</u>

The scenario was implemented with the ns-3 mmWave module which is an extension of the LTE LENA module developed by New York University and the University of Padova for LTE-5G mmWave cellular networks, which implements a dual stack connection for the UE lower layers in [10]. In Figure 1.3.1, the simulation scenario is presented.

The scenario consists in a client and a server interconnected to each other through the LTE-5G network system that integrates the mmWave gNB and mmWave UE radio stacks for dual connectivity.

Each TCP congestion control algorithm (CCA) is analysed based on the congestion window, Round trip time (RTT), Bytes in-flight (BinF), packer delivery ratio (PDR) and the application goodput.

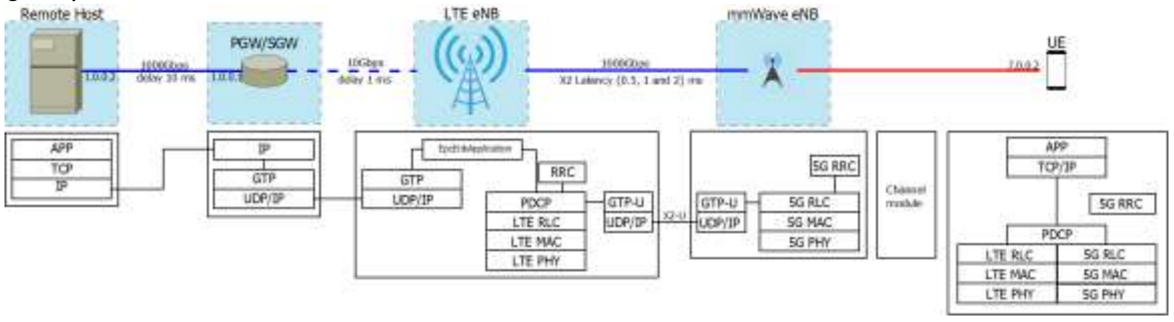


Figure 1.3.1: ns-3 simulation architecture overview.

1.4. Work plan

In order to achieve this goal, the project was divided into three main phases: the study phase, the development phase, and the results and analysis phase.

The overall workflow and schedule of the project is shown in Figure 1.4.1.

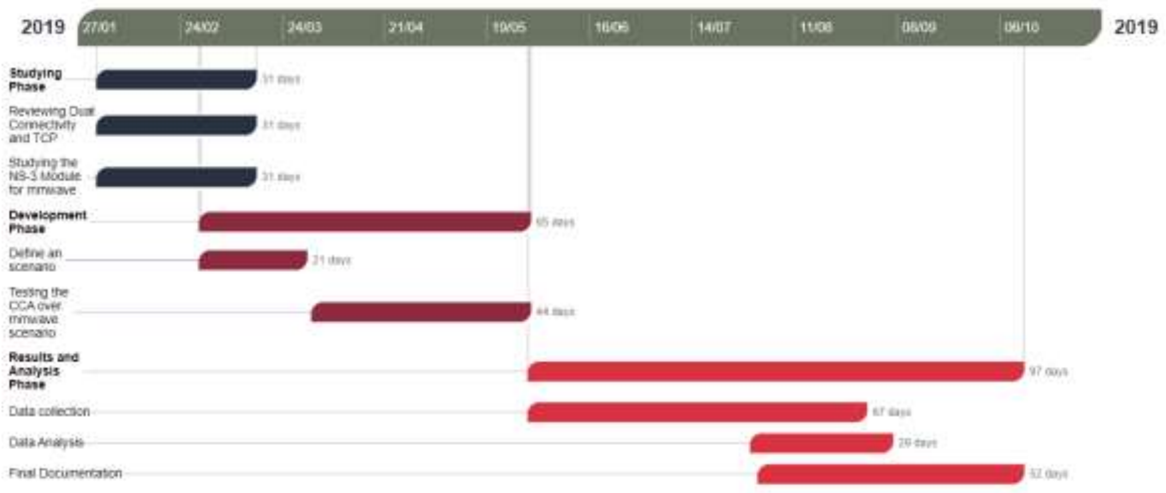


Figure 1.4.1: Thesis project roadmap.

1.5. Report Outline

The structure of the rest of the report is as follows:

In Chapter 2 we give a review the basics of 5G system, Dual Connectivity, Millimetre waves model and transport control protocol concepts, and briefly introduce the network simulator ns-3.

In Chapter 3 we give a detailed description of the solutions proposed to implement Dual connectivity.

In Chapter 4 we describe the simulation framework architecture of our experiments.

In Chapter 5 we present the experimental results of interactions between certain parameters and analyse the outcome performances.

In Chapter 6 briefly explains the budget for the project.

Finally, in Chapter 7 we conclude the report and gives recommendations on further analyses that could be taken.

1.6. Issues Encountered

About Random number generation:

By changing a parameter like the TCP protocol impacts the random number generation, hence you obtain a different channel realization and so different SINR values.

About the mmWave Module issue:

The channel update procedure is the bottleneck of the simulations. To reduce the simulation time, we can increase the channel update time through the attribute "$MmWave3gppChannel::UpdatePeriod$". However, this reduces the accuracy of the simulations, especially if the mobility is relevant.

About the TCP congestion states:

When we get drops and dup ACKs, we go to fast retransmit, and as we recover one dropped packet every RTT i.e. NewReno, the first partial ACK resets the retransmit timer, the subsequent ones do not: This is due to the large buffers, there are still duplicate ACKs in flight, received after the initial RTO. These are causing the TCP state machine to go right back into fast retransmit after the timeout, even though it probably should continue slow start. And somewhere along the line. Worst case scenario!

Storage capacity:

Due to the high number of simulations performed the storage in the computers assigned was not enough. So, I got a Dropbox account and an extra hard drive as a backup.

2. Background

This chapter provides an overview of some concepts in 5G Release 15 and gives a glimpse of potential use cases.

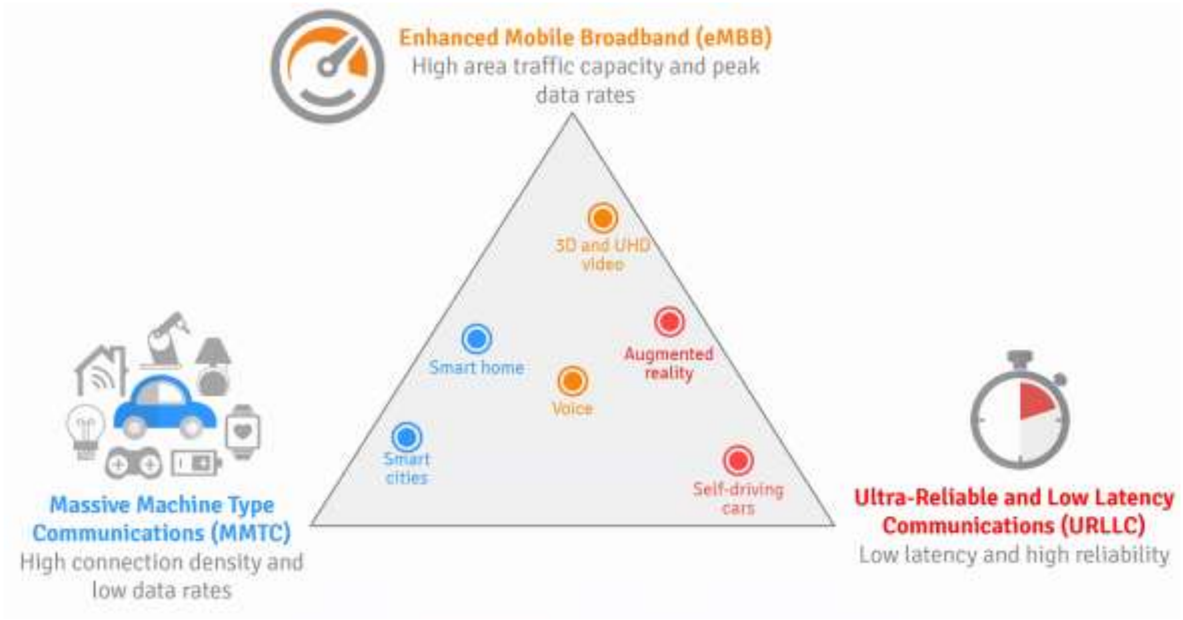
2.1. 5G System

5G is the fifth-generation cellular network technology, which is expected to support very diverse scenarios, such as broadband access everywhere even in ultra-dense areas, as high as 500 km/h user mobility, massive connectivity from all types of low-cost devices, ultralow latency for virtual reality, and ultra-reliability for industry control and vehicle to everything (V2X) [27].

2.1.1. Use cases

There are countless of potential use cases in which 5G can be applied. These can be organized into three main groups by eMBB (enhanced mobile broadband), mMTC (massive machine-type communications) and URLLC (ultra-reliable and low-latency communications) defined by ITU-R.

The enhanced mobile broadband (eMBB) refers to increment of the channel capacity for data rates in the range of Gbps, which involves the deployment of millimeter wave carriers of 28GHz and 39GHz along with sub-6GHz spectrum, spectrum efficiency. The massive machine-type communications refer to the connection of many low power, low cost devices, which have high scalability and increased battery lifetime, in a wide area. Finally, the ultra-reliable and low-latency communications refer to the fact that some applications may involve remote operation and control that will require extreme low latency. Figure 2.2.1 shows the 5g usage scenarios mentioned.



Enhanced Mobile Broadband (eMBB)
High area traffic capacity and peak data rates
3D and UHD video
Smart home
Augmented reality
Voice
Smart cities
Self-driving cars
Massive Machine Type Communications (MMTC)
High connection density and low data rates
Ultra-Reliable and Low Latency Communications (URLLC)
Low latency and high reliability

Figure 2.1.1: 5G usage scenarios (source: Award Solutions)

2.1.2. Service Requirements

These service requirements lead to more challenging performance indicators such as 1 ms transmission delay, 100x higher data rate, and 1000x more connections need to be provided compared to current 4G networks. In Release 15, the study activities for 5G systems standardization were initiated.

The services/applications from the eMBB and URLLC use cases may benefit from the DC technology in order to fulfil the requirements presented, some of these requirements were summarized in [12] and shown in Table 2.1.

KPI	Use case	Target Value
Peak data rate[a]	eMBB	10Gbps for UL
	eMBB	20Gbps for DL
User plane latency	eMBB	4 ms for UL and DL
	URLLC	0.5 ms for UL and DL
	URLLC	1 ms for UL and DL
User experienced data rate	eMBB (dense urban)	50 Mbps for UL
	eMBB (dense urban)	100 Mbps for DL
Mobility[b]	High speed vehicular	120 km/h to 500 km/h
	Vehicular	10 km/h to 120 km/h
	Pedestrian	0 km/h to 10 km/h
	Stationary	0 km/h
Mobility interruption time	eMBB and URLLC	0 ms
Battery life	mMTC	10 to 15 years

[a] The peak data rate is defined as the maximum data rate under ideal conditions (in bps), i.e., under error-free conditions and when all assignable radio resources for the corresponding link direction are utilized. The user experienced data rate is the 5% point of the Cumulative Distribution Function (CDF) of the user throughput.

[b] The maximum UE speed (in km/h) at which a defined Quality of Service (QoS) can be achieved.

Table 2.1: KPIs values expected for 5G networks. Source [12].

2.2. Dual Connectivity

Dual connectivity was introduced in the 3GPP Release 12, where a user equipment (UE) can simultaneously be served by a macro (MeNB) and a small cell (SeNB) operating at different carriers, while the corresponding serving eNBs are interconnected with traditional X2-based backhaul connections (also called non-ideal backhaul), Figure 2.2.1.

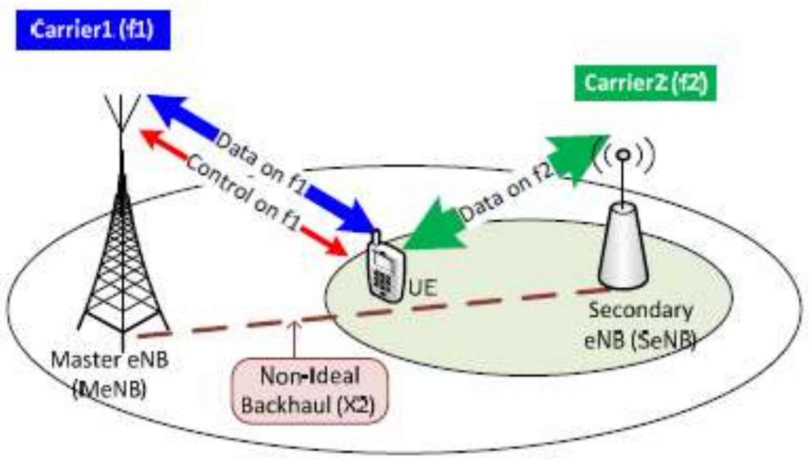


Figure 2.2.1: A typical deployment scenario of Dual connectivity, Source [2].

There are 3 main challenges that were considered to define Dual connectivity: per-user throughput, mobility robustness and increased signalling load due to frequent handover [1].

Dual connectivity stands for simultaneous UE connectivity, with the purpose achieve such a goal in [1] 9 architectures were studied and among them 1A and 3C architectures fit better the challenges mentioned before. The Table 2.2 descripts the benefits and drawbacks encountered. The two solution differ on where the U-plane can be split at either the core or MeNB level, Figure 2.2.2 show the U-plane split solutions.

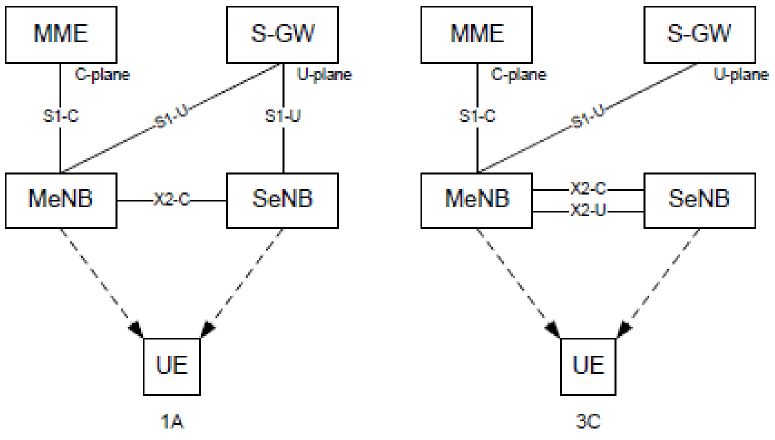


Figure 2.2.2: DC solutions for the split of the U-plane. (Source: Nokia Bell Labs[1])

1A: S1-U terminates in SeNB, independent PDCPs (no bearer split).	**3C: S1-U terminates in MeNB, bearer split in MeNB and independent RLCs for split bearers**
Benefit: The MeNB does not need to buffer or process the packets that come from a bearer that is transmitted by the SeNB. **Drawback:** A UE cannot utilize radio resources across the MeNB and SeNB for the same bearer. Therefore, the user throughput for a given application is not increased by the DC itself.	**Benefit:** A single UE in DC might utilizes radio resources across both MeNB and SeNB for the same bearer, thus increasing the user throughput. **Drawback:** The MeNB needs to route, process and buffer all DC traffic. Another disadvantage is that there must be a flow control between the MeNB and SeNB.

Table 2.2: U-Plane Protocol stack architecture options for DC, [1,2].

In [15] an architectural approach was defined, there are 3-bearer kinds that can be set up, in Figure 2.2.3:

- The MeNB Cell Group (MCG) bearer where the bearer is served using radio resources of MeNB only, Uplink Control Information (UCI) sent via PUCCH on Primary cell (PCell) or PUSCH in other MCG cells.
- SeNB Cell Group (SCG) bearer where the bearer is served using radio resources of SeNB only, UCI sent via PUCCH on Primary SCell (PSCell).
- Split Bearer, there is one flow that is forwarded from the core network to the MeNB PDCP, which splits the traffic into the MeNB RLC and the SeNB RLC.

[1] "Carrier Aggregation and **Dual**. **Connectivity"**. Rapeepat Ratasuk and Amitava Ghosh. Mobile Radio Research Lab, **Nokia** Bell Labs. ISART 2017

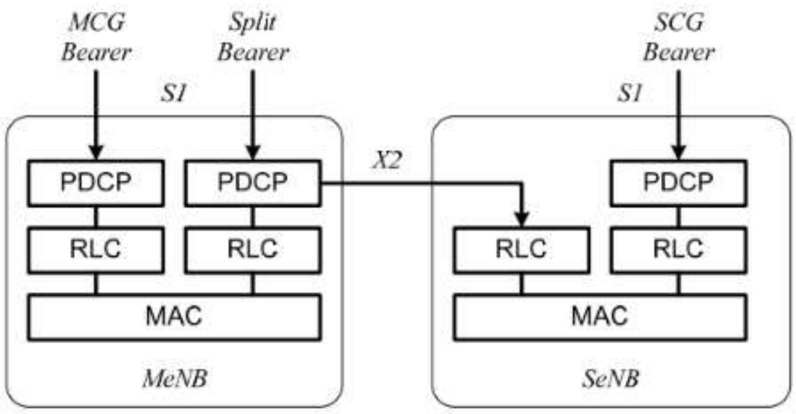
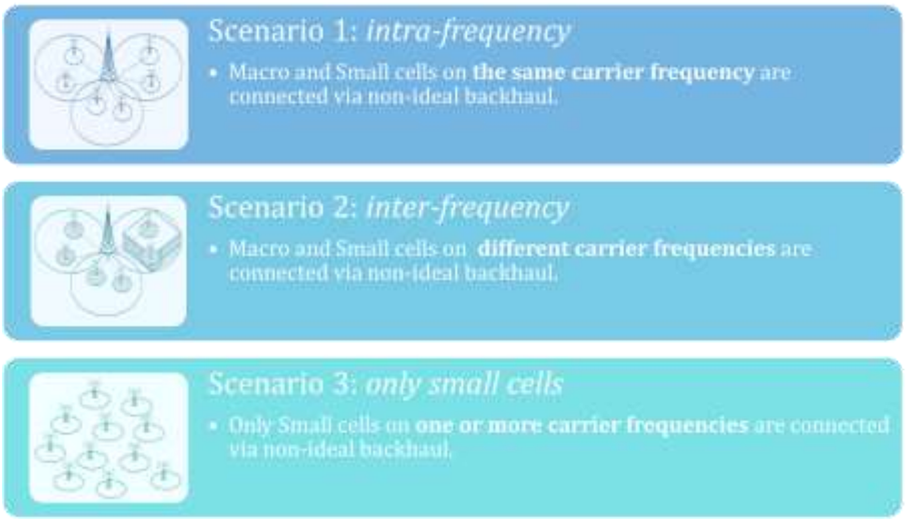


Figure 2.2.3: U-Plane Protocol stack architecture options for DC. (source: 3GPP TS 36.300).

2.2.1. LTE-A Small Cell Deployment with Dual Connectivity

There are 3 small cell scenarios defined for outdoor deployments, as we can see in the figure below, [1].



Scenario 1: *intra-frequency*

- Macro and Small cells on **the same carrier frequency** are connected via non-ideal backhaul.

Scenario 2: *inter-frequency*

- Macro and Small cells on **different carrier frequencies** are connected via non-ideal backhaul.

Scenario 3: *only small cells*

- Only Small cells on **one or more carrier frequencies** are connected via non-ideal backhaul.

Figure 2.2.4: Small cells frequency deployments.

The second scenario was recommended by the 3GPP Release 12 SI (Study Item) to be considered for further studies. However, the scenario 2 presents some challenges, like Mobility robustness for Handover failure (HOF)/ radio link failure (RLF) upon mobility from Pico to macro cells, the UL/DL imbalance between macro and small cells, the increased signalling load due to frequent handover, the difficulty to improve throughput per-user by utilizing radio resources in more than one eNB and the Network planning and configuration.

Section 3 states some solutions proposed and an LTE/5G implementation.

2.2.2. Dual Connectivity in 5G

It is a solution for 5G cellular systems to provide high data rates, throughput, and avoid congestions in the 30-300 GHz band. In comparison with the LTE DC scenario, where a HetNet composed of LTE eNBs operating on different frequencies, another possible solution is proposed for DC.

The new Radio Access Technology (RAT) is defined by 5G as the New Radio (NR). In a DC scenario, the MeNB belongs to LTE RAT and the SeNB to 5G-NR (in which case it is called SgNB). This is a non-standalone solution (in the Appendix A a definition of standalone and non-standalone is defined), where the 5G node and the LTE node are interconnected to the EPC core network of LTE with dual connectivity.

This integration allows multi-RAT connectivity to provide faster mobility and Centralized/Common Radio Resource Management (CRRM) [3]. Moreover, with this integration we can explore RAT diversity to select one or more RATs to establish connection and use transmission diversity to enhance reliability or to increase the per-user throughput.

In the DC architecture, there is a split between the Control and User Planes. The Control Plane controls the transmission of system information and UE connectivity, and the User Plane controls UE specific data, [12].

Connectivity scenarios such as simultaneous connectivity of the Control and User plane to both MeNB and SeNB and a slow procedure like the traditional handover between RATs, were proposed in [10] and [13]. This means the separation of the UE protocol stacks for each node (MeNB and SeNB). Finally, Fast switching was proposed in [14] and [11] where UE would have a Control Plane connection established with two different RATs, but only one of the two is used at a time. Therefore, some coordination between MeNB and SeNB over the X2 interface is needed. This will be explained in Chapter 3.

2.3. Transport Control Protocol

Transmission Control Protocol (TCP) is one of the most used protocols and it is part of the Internet protocol suite. It was defined in the RFC793 to provide ordered and error free bytes transmission between two nodes to reduce packet losses due to congestion.

TCP offers an end to end connection between sender and receiver. It also provides a congestion control (CC) and flow control protocol, which assumes only FIFO queuing. It provides a byte-stream delivery service for applications such as remote login, telnet, hyper-text transfer protocol (HTTP), and file transfer protocol (FTP). Before transmitting data, TCP creates a connection between the source and destination node i.e. three-way handshake. TCP breaks large data into smaller packets and ensures that the data integrity is intact once it is reassembled at the destination node. Besides, it supports error detection or packet loss and retransmit them.

In TCP transmission each packet must be acknowledged by the receiver (ACK). The congestion Window ($cwnd$) determines the maximum quantity of data that can be sent within one round trip time (RTT). The $cwnd$ adaptation procedure depends on the link status.

During TCP slow start phase, the congestion window size increases by one packet per received ACK until:

- The slow start threshold ($ssTresh$) is reached: TCP enters the congestion avoidance phase and the congestion window grows, e.g. NewReno protocol increases by one packet whenever a whole congestion window worth of data has been acknowledged.
- There is a packet loss: if the receiver gets a packet that has a sequence number higher than the expected one, a duplicate ACK counter is triggered. This informs

the sender that the packet with the expected sequence number has been lost and requires retransmission. If three duplicate ACKs are received, TCP assumes that the packet was loss. Thus, it enters in fast retransmit phase by halving the congestion window size and continues in congestion avoidance phase. If the timeout timer is reached (Retransmission Time Out), which is commonly set to 1 second, and no ACKs were received for that packet, TCP reduces the $cwnd$ to one packet and restart from the slow start phase.

2.4. Network Simulator 3 overview

ns-3 is a packet level network simulator that is supported by the open source community for research and educational use. ns-3 aims to build models of computer networks to perform tests. It is written in C++ with optional Python bindings. It contains an LTE module (developed by LENA group) for LTE/EPC simulations scenarios.

ns-3 supports the construction of virtual networks (nodes, channel and applications), event scheduler, topology generation, timers, random variables, etc. Besides, it supports network emulations to interact with real-world software and networks, packet tracing and animation of network simulations.

There is a set of manuals available online, regarding the models design and what to do with them. A good starting point would be the tutorial in [5] and the wiki [7] in order to install ns-3, set up ns-3 scenarios and topologies. The ns-3 Manual [6] will help you to deeper understand the codes e.g. set up of random variables, call back, object model, etc.

2.4.1. Key Concepts

ns-3 has a modular architecture. The src directory contains C++ libraries that implement modular simulation models and network protocols. Later the modules can be instantiated to build a network scenario. Each module has multiple subfolders such as the documentation, the source code of the model itself, helpers, examples and tests.

Below there is a description of the most used folders and subsequently some common terms:

- The helper class meant to ease the setup of other classes, since it hides the complexity involved in setting up a complete scenario. For example, by automatically assigning IP addresses, or connecting the different classes of a protocol stack.
- The model folder is where all the classes of a specific module are.
- Core folder module: Provides the basic structure of the simulator. There are modules for networking protocols, wireless protocols, routing algorithms, mobility, buildings, etc, [4].
- The build folder contains the binaries of the simulator.
- The scratch folder is where you can store your scripts.
- The test folder contains unit testing classes, which are used by a special script, and whose aim is to test ns-3 consistency when we change code.

Common terms:

- $Node$: It's the abstraction of a computing device, represented in C++ by the *class Node*. This class provide methods to configure the device in simulation environments. This means that the node is like a hardware and the methods are peripherals you add.
- $Application$: It's a class that generates some activities to be simulated.
- $Channel$: This class is intended to connect a node to an object, and it works as a communication subnetwork.
- $NetDevice$: It can be the software driver as well as the simulated hardware. A net device is “installed” in a Node in order to enable it to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one Channel via multiple $NetDevices$ [5].

3. State of the art

This chapter presents related research studies involving the interaction and performance of two cells (MeNB and SeNB).

Initially, the idea to connect a user to two nodes was tried to implement in Legacy LTE networks which provide a Carrier Aggregation (CA) technique to combine scattered carriers on different frequencies of those macro and remote small cells within a single eNB. Herein, UEs are controlled by the macro cell in terms of network access, handover, security, and so forth, while user data are delivered by both cells simultaneously. However, this CA is difficult to apply in the mmWave-based network because it needs some coordination between macro and small cells, [24].

In carrier aggregation, the component carriers are managed by the master eNB, i.e. all the information regarding each component carrier is known in real time by the master eNB. Alternatively, Dual connectivity is managed and scheduled by the nodes which can be connected by non-ideal backhaul, this make difficult for the nodes to share information about each component carrier in real time.

In mmWave-based network the LTE eNodeB node performs some coordination among the secondary nodes. Multiple solutions have been proposed to address this issue which are explained in the following section.

3.1. LTE-A Small Cell Networks with Dual Connectivity

This section describes the proposed solution for the connection between the MeNB and SeNB. The flow control, packet scheduling and resource allocation scheme were implemented in the module used the mmWave module. Meanwhile, Delay skew packets flow and load balancing present interesting solutions to guarantee, the time delivery of data to the mobile over multiple transmission paths and to improve the UE's downlink throughput or delay respectively.

3.1.1. Flow Control

A flow control mechanism is proposed in [18] to handle the UE's downlink traffic between the MeNB and SeNB, which are interconnected under different backhaul configurations (e.g. X2 latency, flow control periodicity), in order to guarantee minimum buffering time. There is always data to be scheduled at the small cell eNB.

Moreover, it proposes some enhancements of radio resource management (RRM) functionalities for DC such as UE cell association (i.e. how to configure UEs with DC) and packet scheduling (i.e. how to schedule the UEs configured with/without DC).

Assuming, each eNB has independent radio resources functionalities, the U-plane data flow is split at the MeNB (3C architecture). The MeNB and SeNB operate on different carrier frequencies.

The MeNB periodically determines the available RBs of the SeNB and then sends the amount of data to the SeNB according to the available RBs of the SeNB.

The control flow scheme is described below, Figure 3.1.1 :

- The MeNB forwards an estimated amount of data to SeNB based on UE measurement reports (e.g. CQI) and load conditions (e.g. number of active Users).

- The SeNB starts periodically requesting data to the MeNB based on the average past scheduled throughput per user, the current buffer status, and the pending data forward requests at the SeNB.
- MeNB forwards data to the SeNB only if the buffer size in the MeNB is larger than a certain threshold.
- The SeNB receives the requested data from MeNB, i.e. request-and-forward scheme.

The flow control periodicity is smaller than the backhaul round-trip delay in order to let the MeNB to quickly adapt to the varying channel quality and load conditions in the SeNB.

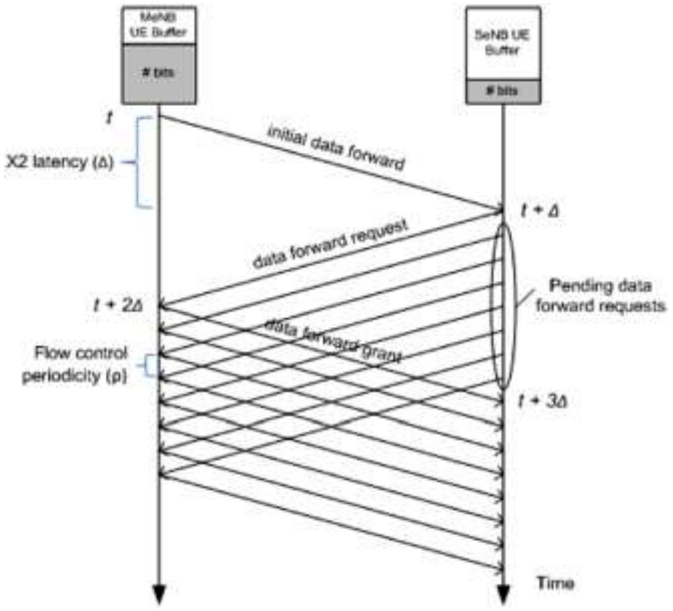


Figure 3.1.1: Schematic illustration of the X2 flow control mechanism, [18].

RRM Enhancements are described in the table below:

UE Cell Association	**Packet Scheduling**
• In LTE, the serving cell is determined based on downlink UE measurements (RSRP on each cell and RSSI on each component carrier). • The UE can be configured to perform measurements of RSRP/RSRQ from its serving and surrounding cells. • It is based on maximum RSRQ value. This means that the serving macro station is selected corresponding to highest received RSRQ from the macro cells, while the serving small station is configured once the received RSRQ from the small cell with the highest received RSRQ from the small cells is above a certain threshold.	• Modified version of the Proportional Fair (PF) scheduler when calculating the scheduling metric: • PF: Allocates users with relative better channel quality. However, when DC and non-DC UEs coexist, it will allocate more resources to users connected to multiple cells (with DC) than users with single connectivity (without DC). “Imbalance problem”. • Cross-carrier PF: It considers the throughput of the DC users over the MeNB and SeNB cells during the resource allocation. • The metric is estimated over a relative longer time window, it doesn’t variate quickly. So, the rate exchange among the involved eNBs can be on a moderate time-scale and is not sensitive to delay over the X2 interface.

Table 3.1: RRM Enhancements description, [18].

Simulation results showed that there is a trade-off between user throughput and SeNB buffering latency, such a trade-off can be properly balanced by configuring the target buffering time in the SeNB and the flow control periodicity, Figure 3.1.2 and Figure 3.1.3.

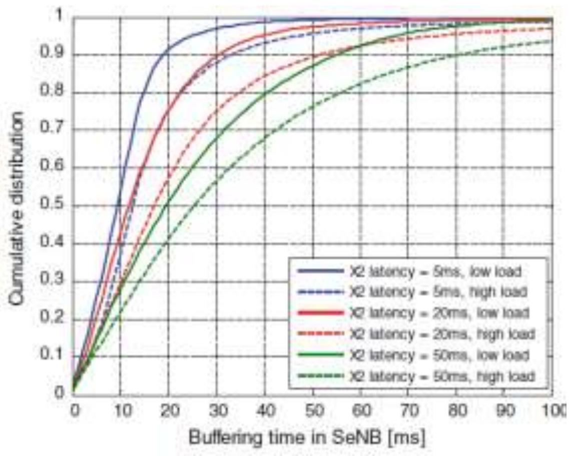


Figure 3.1.2: Buffering time in SeNB with DC under different X2 latencies and traffic loads, [18].

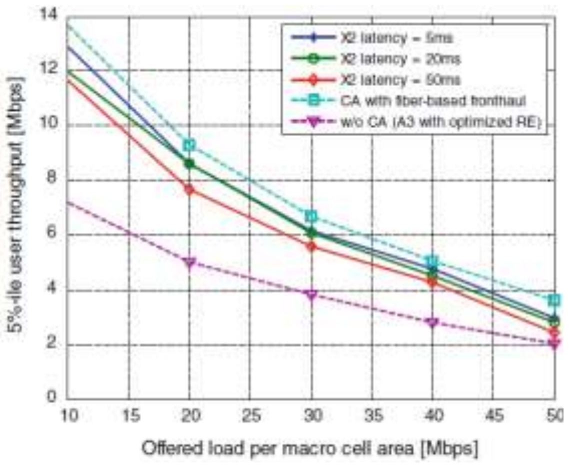


Figure 3.1.3: 5% user throughput with/without DC under different backhaul configurations, [18].

3.1.2. Delay Skew Packet Flow Control

In [19], a data flow controller for 4G and 5G wireless applications is proposed to control the dwell time of the split downlink packets at the wireless transmission nodes. The purpose is to guarantee, the time delivery of data to the mobile over multiple transmission paths within a pre-specified time interval. The packet dwell time of the data in queues of each transmission node is controlled by an inner loop, with the reference dwell times being determined by an outer delay skew control loop.

The outer loop measures the delay skew between the two nodes, from the (Packet Data Convergence Protocol) PDCP layer to the downlink wireless interface. This delay skew measurement along with the sum of the delays are used for control of the reference values of the inner loops, Figure 3.1.4.

The Inner loop controls the dwell time by variation of the data volume of the queues using the downlink backhaul data rate control signals. These queues provide the needed buffering to avoid data starvation.

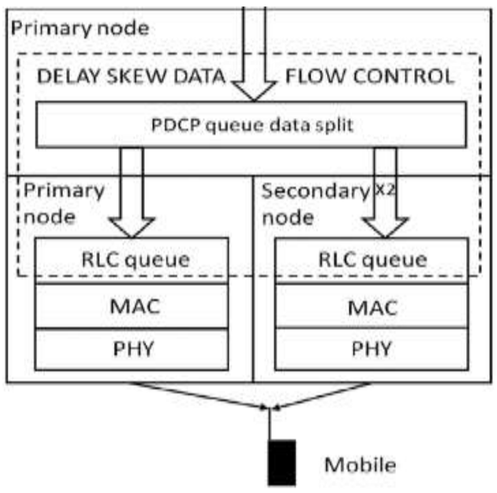


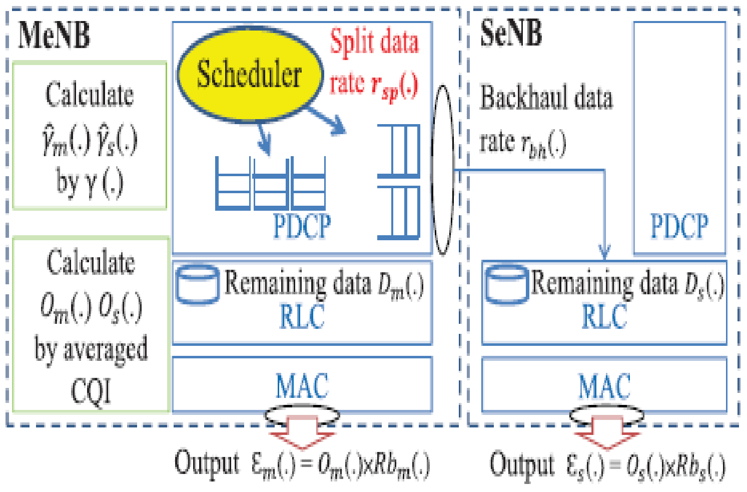
Figure 3.1.4: The dual connectivity architecture. The LTE protocol specifications of the Figure include: PDCP, radio link control (RLC), medium access control (MAC) and physical layer (PHY). The PDCP split bearer architecture of DC release-12, [19].

3.1.3. Packet Scheduling

A Downlink Traffic Scheduling (DTS) mechanism is proposed to maximize the network throughput, in [20]. The MeNB and SeNB operate on different carrier frequencies.

The MeNB periodically arranges the data rate of the DL traffic splitting to SeNBs for UEs (with DC) for an upcoming ‘t’ interval, the Figure 3.1.5 is explain as follow:

- The MeNB imports the collected CQI values and buffer status of the UEs from the SeNBs, a new reporting message for the SeNB is required.
- Then the MeNB estimates the amount of data that can be carried by the UE’s RB through the MeNB an SeNB based on the Adaptive Coding Modulation (ACM) and the UE’s average CQI values (calculated for each UE at the MeNB).
- The MeNB obtains the split data rate by applying the MILP problem.
- The MeNB continually splits and dispatches downlink data to the corresponding SeNB, then the MeNB reset the time.


MeNB
Calculate $\bar{\gamma}_m(.)$ $\bar{\gamma}_s(.)$ by γ (.)
Calculate $O_m(.)$ $O_s(.)$ by averaged CQI
Scheduler
Split data rate $r_{sp}(.)$
PDCP
Remaining data $D_m(.)$
RLC
MAC
Output $\mathcal{E}_m(.) = O_m(.) \times Rb_m(.)$
SeNB
Backhaul data rate $r_{bh}(.)$
PDCP
Remaining data $D_s(.)$
RLC
MAC
Output $\mathcal{E}_s(.) = O_s(.) \times Rb_s(.)$

DTS: Downlink traffic scheduling
NDC: UEs do not have dual connectivity
R_{cbr} : CBR data rate per UE
$F(x)$: Represent the percentage of UE’s traffic will be handled by the MeNB.
$DTS - l$ and $F(70) - l$ represent that there are l SeNBs (l= 1, 2, 4, 8) in the network.
P_{senb} : The transmission power of the SeNB.

Figure 3.1.5: The relationships between parameters and the LTE-A small cell system, [20].

Simulation results shows that the proposed scheme outperform the fixed scheduling method. In a, when varying R_{cbr} we can see that the DTS can perform the best in all cases. In b, when varying values of P_{senb} the DTS performs the best. In c, we can see that DTS outperforms the others in most cases, and $F(100)$ has the worst performance. Finally, in d we can see that DTS outperforms $F(70)$ in all cases. When the R_{cbr} value is smaller, the effects on deploying multiple SeNBs are invisible. On the other hand, when the R_{cbr} value becomes larger, $DTS-8$ can perform the best. But, the $DTS-8$ is only slightly better than $DTS-4$. So, when deploying SeNBs, the network operator must evaluate how many SeNBs are enough with respect to the requirements of the UEs. If the network operator deploys small cells arbitrarily, its capital expenditure (CAPEX) will increase without any gain on network performance, [20].

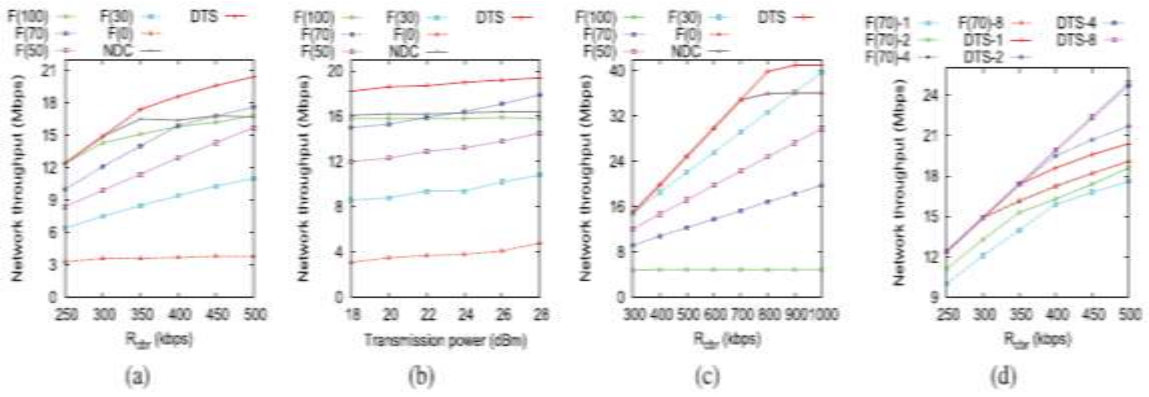


Figure 3.1.6: The simulation results when (a) varying R_{cbr}, (b) varying transmission power of SeNB, (c) placing all UEs nearby the SeNB, and (d) varying the number of SeNBs, [20].

3.1.4. Resource Allocation Scheme

This approach analyses the resource allocation problem i.e. non-ideal flow control between MeNB and SeNB and computational complexity, while considering the ratio of UE with DC, in [21]. The objective problem is to maximize the total utilization of all users. Which is solved by the decomposition theory and generates two sub-optimal problems: achieve maximal resource allocation for macro only UE, and DC Pico UE and legacy Pico UE. The DC Pico UE refers to the DC UE within the small cell region, where its PCell is Macro cell, and the SCell is small cell. The Macro-only UE refers to the legacy UE or DC UE within the macro cell but outside the small cell. Finally, the legacy Pico UE denotes to the legacy UE within the small cell region.

It proposes a distributed resource allocation scheme, which introduces the reserved resource of the macro cell, in order to maximize the system utility and fairness of all users (the DC and the legacy UEs). Figure 3.1.7 gives the flow chart for the coordination procedure:

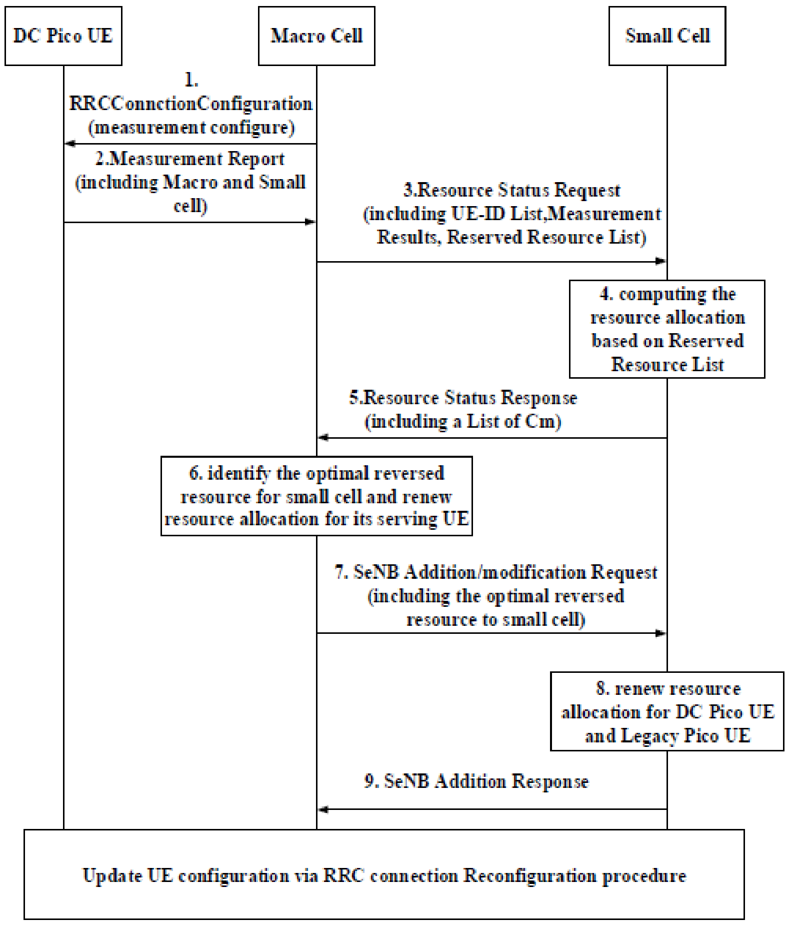
The detailed steps are described as follow, [21]:

A. The macro cell indicates the DC Pico UE to measure the RSRQ of the target small cells and its serving macro cell, respectively.
B. Based on the measurement configuration provided by the macro cell, the DC Pico UE will feedback the measurement result to the macro cell.
C. Considering the small cell cannot acquire the channel information of the DC Pico UE inside its coverage, the Macro cell provides some assistant information to the small cell via Resource Status Request message e.g. UE id list inside the small cell coverage, measurement results of each DC Pico UE.

D. Based on information from the macro cell, the small cell calculates the capacity for each reversed radio resource.
E. The small cell sent the results (Step 4) to the macro cell via the Resource Status Response message.
F. Upon received the feedback from the small cell, the macro cell will calculate the capacity of macro UE. The macro cell selects the optimal resource for the small cell.
G. The macro cell decides to add the SeNB or modify the context for one DC Pico UE. The macro cell will indicate optimal reserved resource to the small cell via UE-associated signalling SeNB Addition Request message.
H. The small cell renew the resource allocation results for DC Pico UE and the legacy Pico UE based on the optimal reserved resource provided by the macro cell.
I. The small cell response the macro cell about the new resource configuration is applied.

The solution derives into a distributed signalling scheme between macro cell and small cell. Which enables a faster scheduling between two nodes. The macro and Pico cell operate on different carrier frequencies with 3C U-plane.

Simulations show improvement of the fairness for different types of UEs and reduces the amount of exchanging information between macro cell and small cell.


DC Pico UE
Macro Cell
Small Cell
1. RRCConnctionConfiguration (measurement configure)
2.Measurement Report (including Macro and Small cell)
3.Resource Status Request (including UE-ID List,Measurement Results, Reserved Resource List)
4. computing the resource allocation based on Reserved Resource List
5.Resource Status Response (including a List of Cm)
6. identify the optimal reversed resource for small cell and renew resource allocation for its serving UE
7. SeNB Addition/modification Request (including the optimal reversed resource to small cell)
8. renew resource allocation for DC Pico UE and Legacy Pico UE
9. SeNB Addition Response
Update UE configuration via RRC connection Reconfiguration procedure
Figure 3.1.7: The flow chart for the coordination procedure.

3.1.5. Load Balancing and Splitting Bearer

This approach evaluates the performance of load balancing (LB), and splitting bearer (SB) for Dual Connectivity (DC) in order to improve the UE’s downlink throughput or delay, in [22]. The table below briefly describes each of them:

Load balancing (LB)	Splitting bearer (SB)	Smart SB
Bearer switching from a macro cell to a Pico cell. This means that the unsatisfied users move to Pico cell with the highest SINR (the target cell). A check is performed for sufficient spare resource in the target cell to ensure user satisfaction.	Link aggregation. Allocate unsatisfied users in the target cell to ensure user satisfaction. If there is insufficient resource to meet this objective all available resource will nevertheless be taken.	Identifies the users that are satisfied and located in cells where the load is high (RB load is 100%) and other users in the same cell that are not satisfied. The satisfied users may then have their bearers split, freeing up resource for the unsatisfied users.

Table 3.2: Description of Load balancing (LB), Splitting bearer (SB) and Smart SB.

Since the schedulers at the macro and Pico cell are weakly coupled (specially for cell edge users), the simulation is observed at an arbitrary instant of time through "static Monte Carlo Simulation". In the Outdoor scenario, when assigning users according to a full network loading (baseline), users in overloaded cells are unable to meet the minimum video rate requirement. When both algorithms target weakly performing users, they obtain similar performance, with a marked improvement in the user satisfaction rate versus the baseline.

The figure below shows the capacity enhancements provided by LB and SB for the associated user satisfaction rates. After LB and SB, the network can serve users with a target rate requirement that is ~2.3 times higher than the baseline, while maintaining the user satisfaction,95%. Activating SB on top of LB gives a bit extra gain.

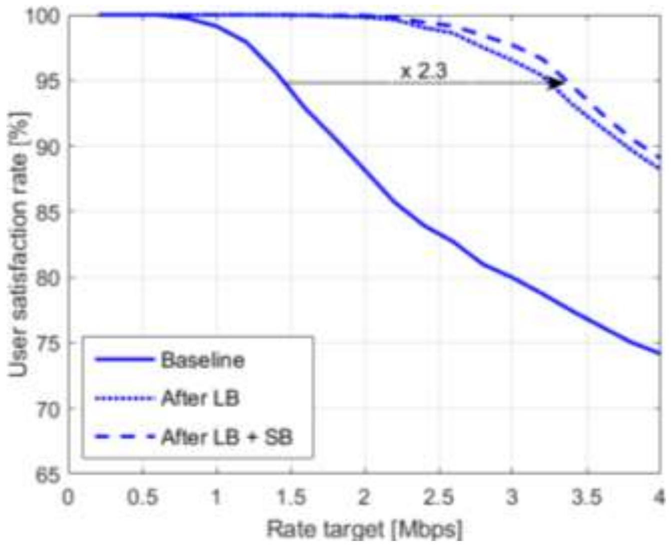


Figure 3.1.8: User experience performance for the indoor & outdoor scenario with 3 buildings per macro and 70% indoor UEs

3.2. LTE-5G Tight Integration

This was proposed in [13], which implements dual connectivity protocols that enable mobile devices to maintain physical layer connection to 4G and 5G cells simultaneously. The main procedures are described since the simulation model of this work is used in this thesis for the study of TCP protocols.

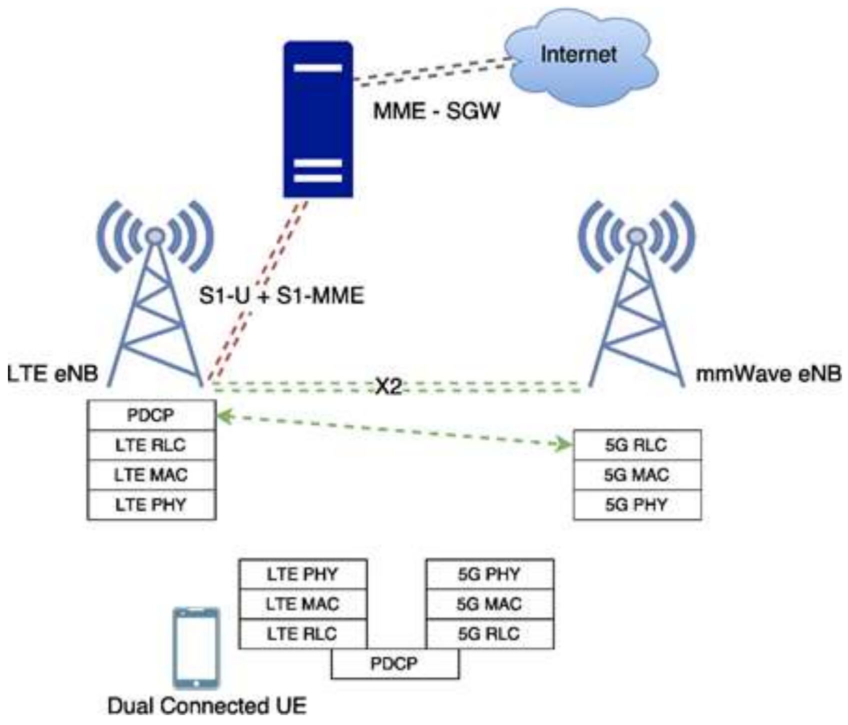


Figure 3.2.1: LTE-5G tight integration architecture, [13].

The Figure 3.2.1 shows the proposed architecture in the LTE-5G integration [13], where the two eNBs are connected via an X2 link. Each LTE eNB coordinates a cluster of mmWave gNBs (in the Appendix A we defined the type of nodes for 5G). The coordinator may also be placed in a new node at the core network or can be based on Network Function Virtualization (NFV, Appendix A).

A. C-Plane for Measurement Collection

The UE broadcasts Sound Reference Signals (SRS) for each direction in dedicated slots. Each cell scans its angular directions and monitors the strength of the received SRS. Then, the mmWave gNBs fills a Report Table (RT) with the highest SINR and its best direction for each UE and sends it to the coordinator. The coordinator collects the RTs coming from all the mmWave gNBs and builds a Complete Report Table (CRT) for each UE, based on the RT received. The optimal eNB and direction for each UE is then selected considering the SINR for each (mmWave gNB, direction) pair. Finally, the LTE eNB reports to the UE which is the pair that yields the best performance.

The LTE RRC is chosen because it offers higher stability and reliability over millimeter wave radio access. The coordinator notifies to mmWave gNB about the optimal direction to steer the beam for serving each user.

B. U-Plane (PDCP Layer Integration)

For each bearer, the user plane has an instance at the LTE eNB PDCP/RLC layer and mmWave gNB RLC layer.

The packets are routed from the S-GW to the LTE eNB, which route them to either the local or remote RLC layer. It allows a non-co-located deployment, since the synchronization among the lower layers is not required.

The LTE connection is used as a backup. Although, mmWave links are greater than LTE links in capacity, LTE eNBs will serve more users than the mmWave gNBs. Users will be connected to LTE eNB when mmWave gNB is in outage.

The integration at the PDCP layer ensures ordered delivery of packets to the upper layers, which is useful in handover operations.

C. Fast Switching (FS)

The Fast Switching mechanism supports handovers from LTE RAT to mmWave RAT eNB or mmWave RAT eNB to LTE RAT, when all mmWave gNBs are in outage. For FS, an RRC message is sent to the UE on the LTE link, and also a notification to the mmWave gNB is sent via X2 if the switch is from mmWave to LTE, to forward the content of the RLC buffers to the LTE eNB.

It allows an uninterrupted connection to the LTE anchor point, as shown in Figure 3.2.2 and Figure 3.2.3.

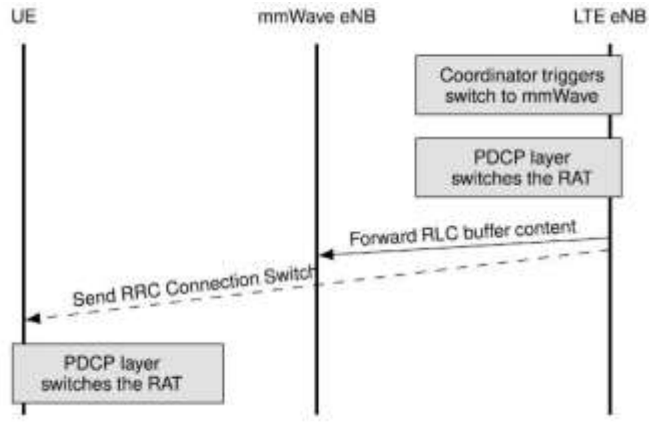

UE
mmWave eNB
LTE eNB
Coordinator triggers switch to mmWave
PDCP layer switches the RAT
Forward RLC buffer content
Send RRC Connection Switch
PDCP layer switches the RAT

Figure 3.2.2: Switch from LTE RAT to mmWave RAT, [13].

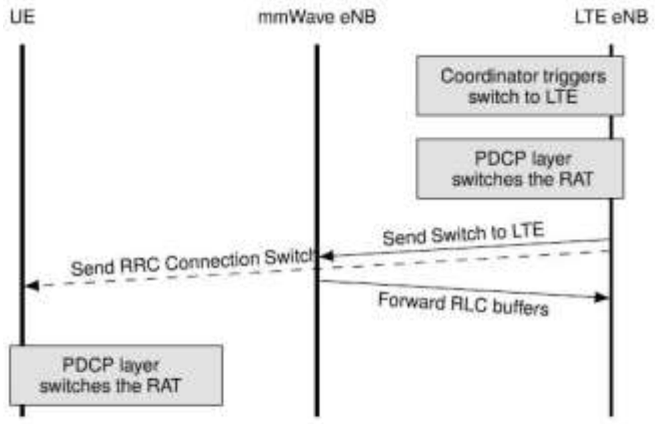

UE
mmWave eNB
LTE eNB
Coordinator triggers switch to LTE
PDCP layer switches the RAT
Send Switch to LTE
Send RRC Connection Switch
Forward RLC buffers
PDCP layer switches the RAT

Figure 3.2.3: Switch from mmWave RAT to LTE RAT, [13]

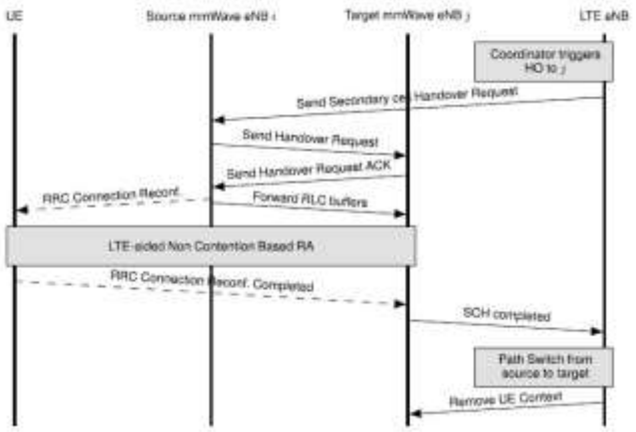

UE
Source mmWave eNB i
Target mmWave eNB j
LTE eNB
Coordinator triggers HO to j
Send Secondary cell Handover Request
Send Handover Request
Send Handover Request ACK
RRC Connection Reconf.
Forward RLC buffers
LTE-aided Non Contention Based RA
RRC Connection Reconf. Completed
SCH completed
Path Switch from source to target
Remove UE Context

Figure 3.2.4: Secondary cell Handover procedure (SCH), [13]

D. Secondary cell handover (SCH)

SCH occurs from a secondary mmWave gNB to a different mmWave gNB, Figure 3.2.4.

The measurement collection procedure allows the coordinator to define the best beam. The LTE eNB triggers the procedure to setup a new RLC.

It is faster than a standard intra RAT handover (the core network is not involved).

E. Secondary Cell HO decision method

The algorithm considers the Report Tables (RTs) and a threshold in time (time-to-trigger, TTT). When a better SINR than the current one appears, the coordinator checks for TTT seconds. If the condition still holds, it triggers the SCH. However, if during the TTT seconds interval, the SINR of third cell becomes better than the target cell by less than 3 dB, the handover remains scheduled for the original target eNB and the current SINR becomes the highest, and if the original cell SINR becomes the highest, then the SCH is cancelled. Finally, the TTT can be a fixed value or dynamically adapted so when the SINR between cells is high, the TTT must be small or vice versa.

4. Methodology and Project Development

This chapter provides the basic information about the simulation framework used in this project with the configurations of LTE-5G Dual connectivity along with the channel model development based on the 3GPP TR 38.900 Release 14 specification.

The protocols chosen to test TCP with Dual connectivity over millimeter waves are NewReno, Hybla, Scalable, HighSpeed, Cubic, Illinois and HTCP. These codes can be found in the following link: https://github.com/nyuwireless-unipd/ns3-mmwave/tree/02e87e5f6276f56c75343f45f41aa2376682f3f9/src/internet/model.

The following subsections describe the network topology, system architecture and Congestion control algorithm implemented.

4.1. Network Configuration

In order to test the behaviour of each Protocol in the network, we used a simulation topology consisting in a client and a server application interconnected to each other through the LTE-5G network system, as shown in Figure 4.1.1, which represent the mmWave gNB and UE radio stacks, respectively, along with a outlook on the end-to-end structure of the simulator.

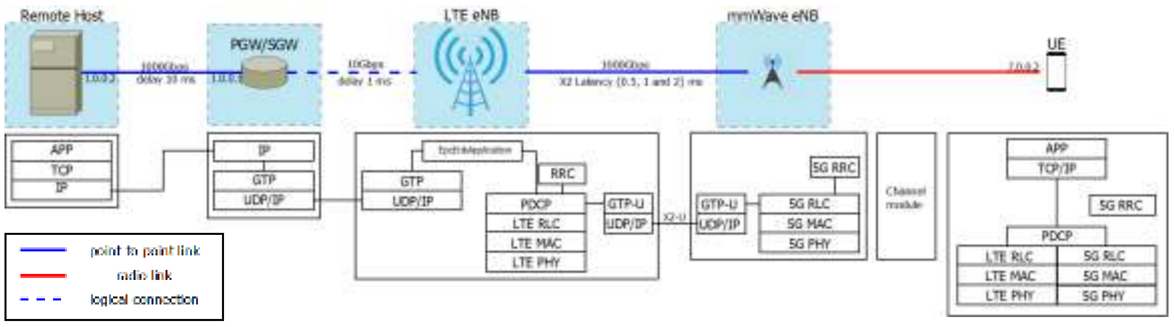
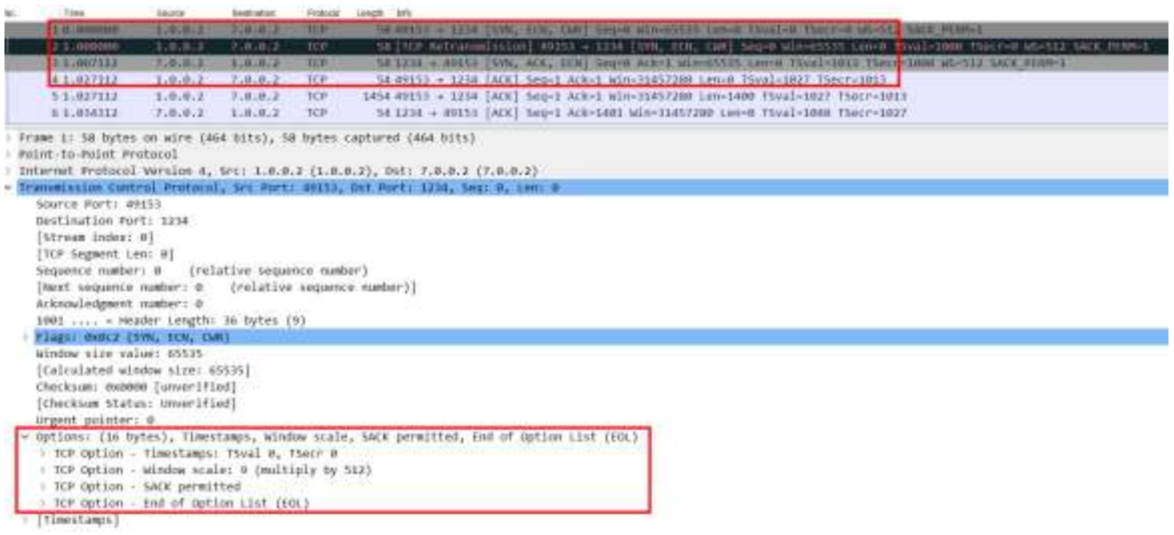


Figure 4.1.1: ns-3 topology of the mmWave module.

Each TCP congestion control algorithm is analysed based on the congestion window, Round trip time, Bytes in-flight and the application goodput.

From Figure 4.1.2, we can verify the TCP options configured, and the initial synchronization (3-way handshake). Where IP 1.0.0.2 represent the server address and the 7.0.0.2 corresponds to the client. In the figure we can also notice that the frame 2 is a TCP retransmission of the first synchronization message, this is because the first frame (SENDER to RECEIVER) was sent right at the beginning of the simulation and the client wasn't connected yet. So, after the timer runs out (1 second) it sends again the message.



```
No.  Time       Source   Destination  Protocol  Length  Info
   1 0.000000   1.0.0.2  7.0.0.2      TCP         58 49153 → 1234 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 TSval=0 TSecr=0 WS=512 SACK_PERM=1
   2 1.000000   1.0.0.2  7.0.0.2      TCP         58 [TCP Retransmission] 49153 → 1234 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 TSval=1000 TSecr=0 WS=512 SACK_PERM=1
   3 1.007313   7.0.0.2  1.0.0.2      TCP         58 1234 → 49153 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 TSval=1013 TSecr=1000 WS=512 SACK_PERM=1
   4 1.027312   1.0.0.2  7.0.0.2      TCP         54 49153 → 1234 [ACK] Seq=1 Ack=1 Win=31457280 Len=0 TSval=1027 TSecr=1013
   5 1.027112   1.0.0.2  7.0.0.2      TCP       1454 49153 → 1234 [ACK] Seq=1 Ack=1 Win=31457280 Len=1400 TSval=1027 TSecr=1013
   6 1.034312   7.0.0.2  1.0.0.2      TCP         54 1234 → 49153 [ACK] Seq=1 Ack=1401 Win=31457280 Len=0 TSval=1040 TSecr=1027

> Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
> Point-to-Point Protocol
> Internet Protocol Version 4, Src: 1.0.0.2 (1.0.0.2), Dst: 7.0.0.2 (7.0.0.2)
v Transmission Control Protocol, Src Port: 49153, Dst Port: 1234, Seq: 0, Len: 0
    Source Port: 49153
    Destination Port: 1234
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1001 .... = Header Length: 36 bytes (9)
  > Flags: 0x0C2 (SYN, ECN, CWR)
    Window size value: 65535
    [Calculated window size: 65535]
    Checksum: 0x0000 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (16 bytes), Timestamps, Window scale, SACK permitted, End of Option List (EOL)
    > TCP Option - Timestamps: TSval 0, TSecr 0
    > TCP Option - Window scale: 9 (multiply by 512)
    > TCP Option - SACK permitted
    > TCP Option - End of Option List (EOL)
  > [Timestamps]
```

Figure 4.1.2: Wireshark capture of the TCP options and handshake

4.2. System model

In this section the modules used for the implementation are explained. The Dual connectivity module contains the dual stack lower layers for the UE and mmWave gNB.

In order to evaluate the performance of Dual connectivity over mmWave four different environments have been used. Most of the work described in this chapter has been carried out over the simulated environment and for comparison purposes the findings and results have been correlated with the behaviour in the other two deployments. Since the configuration and explanation of the simulated environment is comprised of many parameters and in order to help the reader understand the setup, Table 2 gathers the most important information about the simulation environment regarding the configuration parameters and experiment-related conditions. As the simulated environment, ns-3 simulator with the LTE capabilities of LENA module is used.

In order to simulate the link to the server, the propagation delay between the remote host and the Packet Data Network Gateway (PGW) was set up at 10 milliseconds.

The Radio Link Control (RLC) layer was set at the Acknowledged Mode (AM) in order to resemble the most commonly deployed configuration in real-world.

Regarding the radio resources, the 5G Nodes are configured in the model to have a value of 72 available physical resource blocks (PRB) at 28 GHz frequency band.

4.2.1. ns-3 Dual Connectivity module

The ns-3 mmWave module used is an extension of the LTE LENA module developed by New York University and the University of Padova for LTE-5G mmWave cellular networks, which implements a dual stack connection for the UE lower layers, in [10]. The implemented module allows the UE to be simultaneously connected to both RAT and uses the LTE eNB as U-plane backup making possible the evaluation of end-to-end performance.

The module is shown in Figure 4.2.1. The diagram divides the module into four main blocks, the $McUeNetDevice$, the $LteEnbNetDevice$, $MmWaveEnbNetDevice$ and Channel block.

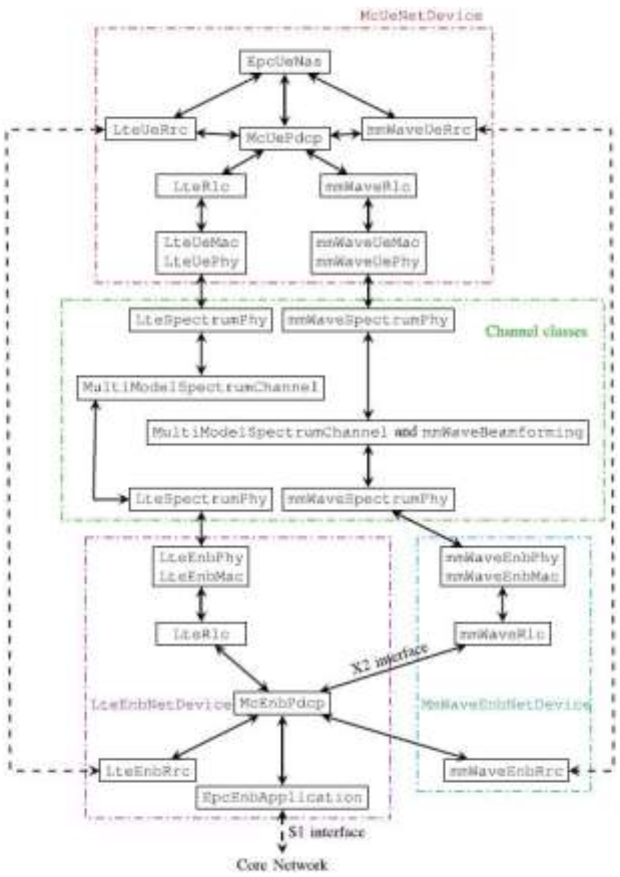


Figure 4.2.1: Block diagram of a dual-connected device, an LTE eNB and a mmWave gNBs, [11].

The $McUeNetDevice$ contains a dual stack classes for mmWave and LTE lower layer and methods to forward packets to the TCP/IP stack. The dual connected UE is associated to a single $EpcUeNas$, but with a separate LTE and mmWave PHY and MAC layers. The $McUeNetDevice$ instances the RRC layer for both links to exchange information between them. The LTE RRC manages both the LTE connection and features related to DC, while the mmWave RRC handles only the mmWave link (avoiding latency in control commands).

The classes $McEnbPdcp$ and $McUePdcp$ classes represent the eNB PDCP and UE PDCP layers. Besides, the bearer is split at PDCP layer of the LTE eNB (local coordinator) which sends the packets through the X2 link between eNBs (i.e., EpcX2), to the mmWave RLC layer, and vice versa in uplink.

The Channel box represents the LTE and mmWave channels models. They are configured independently since they belong to different frequency systems thus there is no interference between them.

4.2.2. ns-3 mmWave channel model

It implements a channel model, a custom PHY and MAC layers for 5G mmWave protocol stack and relies on the LTE module (LENA),[9], for the upper layers.

Channel Modelling

There are 3 channel models available in the mmWave module. The 3GPP Statistical Channel Model which is based on the official 3GPP channel model for 6-100 GHz frequency band, [10]. It defines different scenarios for cellular network deployments: urban, rural and indoor. The Ray-Tracing or Measurement Trace Model based on traces from measurements or third-party ray-tracing software. And the third is the NYU Statistical Model that presents two pathloss models to capture the LOS and NLOS condition.

The simulations are carried out using the 3GPP statistical model because it provides multiple optional features that can be plugged into the simulations e.g. the simulations are implemented for urban microcell scenario deployment (UMi).

Error Model

Based on the ns-3 LENA module, it allows to map the SINR to an error probability for the whole transport block (TB), considering modulation and coding techniques. In case of error the PHY doesn't forward the incoming packets to the upper layer. Instead, it triggers a retransmission process.

Interference

In the case of directional mmWave signals, the interference is less significant. Because they are assumed to be power limited.

Intra cell interference is considered in cases of Spatial Division Multiple Access (SDMA) and Multi-User MIMO, where users are multiplexed in the spatial dimension but operate in the same time-frequency resources.

Thus, the interference computation scheme that considers the beamforming vectors associated with each link, is included in the $MmWaveInterference$ *class*, [11].

PHY Layer

It implements a TDD frame and a subframe structure, like TDD-LTE, but is more flexible in allocating control and data channels within the subframe and is suitable for variable TTI MAC scheme, [11].

The $MmWaveEnbPhy$ and $MmWaveUePhy$ classes model the physical layer for the mmWave gNB and the UE, respectively. They oversee the transmission and reception of physical control and data channels, simulate the start and end of frames, subframes and slots, and finally deliver decoded control and data packets to the MAC layer.

MAC Layer

Implemented in the $MmWaveEnbMac$ and $MmWaveUeMac$ classes, they are intended to coordinate procedures such as scheduling and retransmission. Moreover, they interact with the RLC layer to receive periodic reports on the buffer occupancy, i.e., the Buffer Status Reports (BSRs), and with the physical layer classes for the transmission and reception of packets. To carry out their functionalities, the MAC classes interact with other classes e.g. Adaptive modulation and coding (AMC), Hybrid ARQ Retransmission and Schedulers.

RLC Layer

This is based on the LTE RLC entity, but the RLC AM retransmission is modified to be compatible with the mmWave PHY and MAC layers. It has the capacity to segment and retransmit in order to support intermittent mmWave channel. Also, an Active Queue Management (AQM) for the RLC buffer is used to avoid congestion.

4.2.3. TCP in ns-3

ns-3 implements TCP by using several classes that provide reliable transport protocol services and communicate with the network layer. The current standard release contains multiple TCP variants where NewReno is defined as the default congestion control algorithm in class $TcpCongestionOps$. Algorithms like Hybla, Highspeed, Scalable, etc, are pluggable components implemented as child classes of $TcpNewReno$.

The main classes that implement the TCP functionalities are:

- $TcpSocketBase$ class inherits from $TcpSocket$ and provides the interface between the application layer and TCP sockets, and is the base for the different TCP congestion control (CC) variants.
- $TcpSocket$ class is an abstract class that contains the attributes of TCP socket like $SenBufSize$, $RcvBufSize$ and, $SegmentSize$ (buffer size for sending and receiving).
- $TcpHeader$ class contains the implementation of the TCP header fields such as the port number, sequence and ACK numbers.
- $TcpTxBuffer$ class provides a buffer service to the application layer, which allows to buffer until the ACK arrives.
- $TcpRxBuffer$ class provides a buffer for incoming packets to be reordered.
- $TcpL4Protocol$ class provides an interface between the TCP socket and the lower layers, and it is responsible of checksum.
- $TcpSocketState$ class tracks some attributes like the congestion state, congestion window and slow start threshold for each connection.

A. Congestion State machine

The connection establishment in TCP is set up by the well-known three-way hand-shake procedure, where the maximum segment size (MSS) is agreed at this point. Once the end users are set, the slow start phase starts (a variant of this protocol is the hybrid slow start which is implemented in Cubic) and the $cwnd$ increases by one packet per received ACK until the $ssTresh$ is reached or there is a congestion event (e.g. packet loss or packet disorder). If the $ssTresh$ is reached, TCP enters the congestion avoidance phase and the $cwnd$ grows according to the strategy of the congestion algorithm implemented.

Based on the NS-3 implementation, the congestion state machine is devised as in Figure 4.2.2. Here OPEN means that the TCP is in Normal state (no dubious events), DISORDER state is when the sender receives duplicate ACK, if the sender receives more than 3 duplicate ACK the RECOVERY is triggered and $cwnd$ is reduced. The LOSS state is when time runs out. These states are mapped in the simulator as follow:

- CA_OPEN: 0
- CA_DISORDER: 1
- CA_RECOVERY: 3
- CA_LOSS: 4

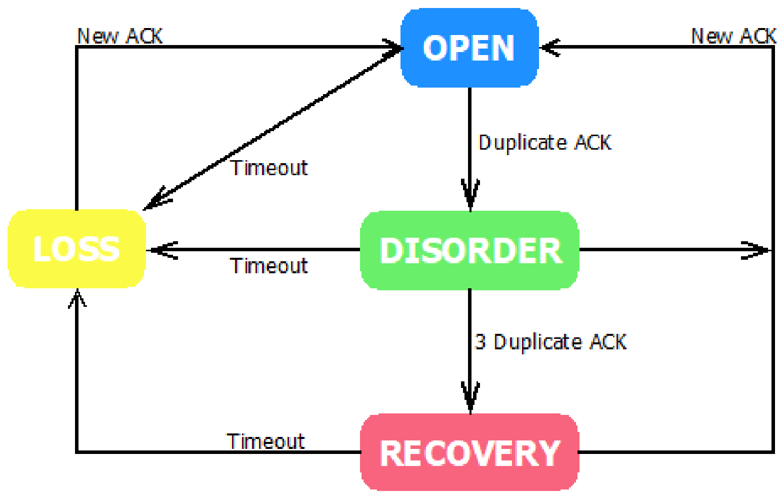


Figure 4.2.2: ns-3 congestion state model.

B. Slow start and congestion avoidance

In the phase of a slow start and congestion avoidance algorithm implementation must be used by a TCP sender data controlling to inject outstanding data into the network. This can be handled by two variables the congestion window ($cwnd$), this is a sender side limit of data amount transmission before receiving an acknowledgment (ACK) and receiver advertised window (RWND) is the receiver side limit of data amount outstanding for processing. Another variable is slow to start threshold ($ssThresh$), it is used for determining the congestion avoidance algorithm is used to data transmission control. The initial $cwnd$ size depends on the sender maximum segment size ($SMSS$) and $cwnd$ set to less or equal to $2 \times SMSS$. on receiving every ACK $cwnd$ is increased by full segment size until not detected duplicate ACK. when packet loss detected then set the maximum $ssThresh$ by the maximum value between $inFlight/2$ and $2 \times SMSS$, where flightsize is the amount of outstanding data in the network. The threshold selection is important for quality of service requirement for the communication network.

C. Fast Retransmit/Fast Recovery

Fast retransmit algorithm reduces the time the sender waits before retransmitting a lost segment, it counts the incoming duplicate ACKs till it receives 3 duplicate ACKs, ($ReTxThreshold$ attribute). Then the TCP performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire.

1. If $3\ dupACK$ is received, set $ssThresh$ to $\max(flightsize/2, 2 \times SMSS)$

2. Retransmit the lost segment and set $cwnd$ to $ssThresh + 3 \times SMSS$.

3. For each additional duplicate, ACK received, increment $cwnd$ by SMSS.

4. Transmit a segment, if allowed by the new value of $cwnd$ and the $rwnd$.

5. When the next ACK arrives that acknowledges new data, set $cwnd$ to $ssthresh$ This ACK should be the acknowledgment elicited by the retransmission from step 1, one RTT after the retransmission, [8].

4.3. Congestion control Algorithms

4.3.1. TCP New Reno

The NewReno [23] protocol is a packet loss-based algorithm which employs the additive increase multiplicative decrease (AIMD) mechanism and it is implemented as part of the class $TcpCongestionOps$. NewReno modified TCP Reno to detect multiple packet losses by entering fast-retransmit when it receives multiple duplicate packets (3 duplicate acknowledged packets). NewReno's states diagram is shown in Figure 4.3.1, it describes how NewReno when in slow start phase the $cwnd$ grow by one packet per acknowledgement reception then it moves from slow start phase to either congestion avoidance in case it reaches the threshold value or Fast-Retransmission phase in case of congestion events occurs before. As we can see during congestion events it doesn't exit Fast-Recovery phase until all the data, which was out standing at the time, is retransmitted and acknowledged. If all missing the segments which were outstanding when we entered Fast-Recovery are acknowledged, then it exits Fast-Recovery phase and sets the congestion window ($cwnd$) to the slow start threshold ($ssThresh$) and continues congestion avoidance. Otherwise, if the ACK is a partial ACK then it deduces that the next segment in line was lost and it re-transmits that segment and sets the number of duplicate ACKs received to zero. It exits Fast recovery phase when all the data in the window is acknowledged.

During congestion avoidance (CA), the congestion window is incremented by roughly 1 full-sized segment per round-trip time (RTT), as follow:

$INC = max\left\{1\,, \frac{SMSS \times SMSS}{cwnd}\right\} bytes$, where INC is the congestion window increment during congestion avoidance, SMSS is the sender maximum segment size ($SMSS = 1400 bytes$) and $cwnd$ is the congestion window, e.g. if $SMSS = 1400 bytes$ and $cwnd = \ 2\ Mbytes$ the increment would be the integer value of $max\left\{1\,, \frac{1400 \times 1400}{2 \times 1024 \times 1024} = 0.93\right\} byte$. this means that for large $cwnd$ the default increment value is 1Byte and for small $cwnd$ the INC is greater than 1 Byte.

While in Fast-Recovery phase, the $ssThresh$ is set to $\max\left\{2\,SMSS,\ \frac{inFlights}{2}\right\}$, where $inFlights$ is the number of bytes unacknowledged. This means that the flow of packets is reduced to the maximum value between 2 SMSS and half of the numbers of Bytes in-flight when a congestion event occurs.

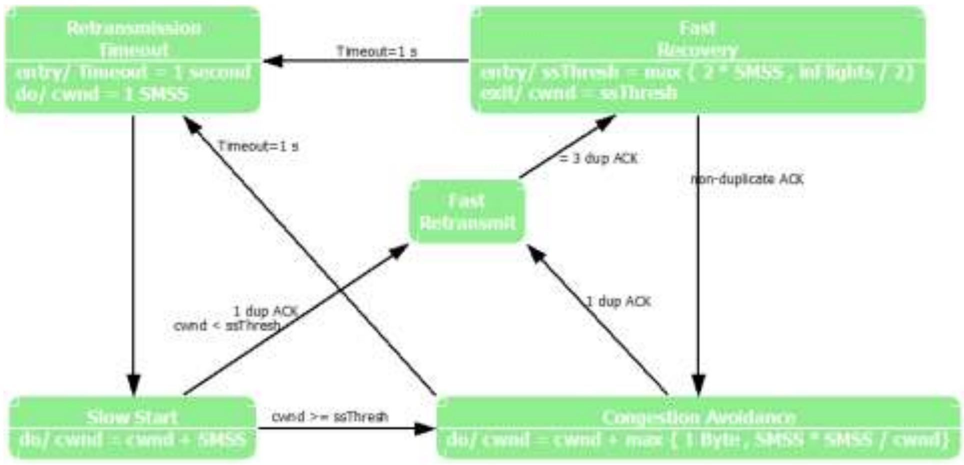


Figure 4.3.1: TCP NewReno's states diagram in ns-3.

The simulations enabled SACK option so in recovery phase the $cwnd$ is set to $cwnd = ssThresh + (dupAckCount \times SMSS)$. Also, during recovery phase, the congestion window size is kept the same.

4.3.2. TCP Hybla

Hybla is a packet loss-based algorithm and the $TcpHybla$ class is derived from NewReno's class, it was proposed to work on links with long delays. Hybla's states diagram is shown in Figure 4.3.2. This algorithm aims to reduce the RTT dependency (detach the growth of congestion window from RTT), the algorithm introduces three variables; a minimum RTT, a reference RTT and a rho parameter that normalizes the RTT by:

$rho = \max\{\frac{\min RTT}{reference\ RTT}, 1\}$, where the $reference\ RTT$ is set up to 50 ms

During the slow start phase, the congestion window grows by:

$INC = \min\{SMSS \times (2^{rho} - 1), ssthresh\}$,

In Congestion avoidance, the $cwnd$ grows by:

$INC = SMSS * \left(\frac{rho^2}{cwnd}\right)$,

In congestion events Hybla follows NewReno's procedure but during recovery phase the congestion windows is set to $sstresh + dupAckCount * SMSS$ (SACK option enabled), Hybla's diagram is shown in Figure 4.3.2. The objective of Hybla is to compensate the long RTT, a flow with high delay using TCP Hybla has higher throughput than the same flow using standard TCP.

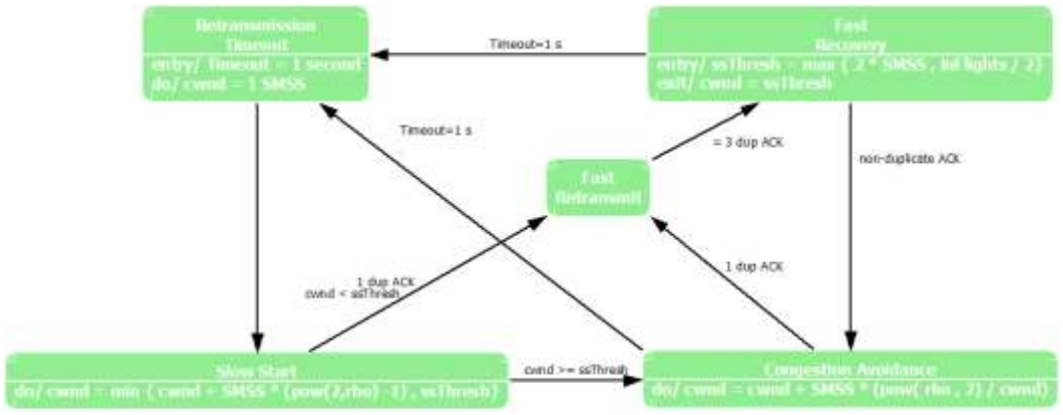


Figure 4.3.2: TCP Hybla's states diagram in ns-3.

4.3.3. TCP Scalable

TCP Scalable algorithm is also derived from NewReno's class, it aims for high bandwidth delay products (BDP) as it uses packet loss as a feedback. TCP Scalable's states diagram is shown in Figure 4.3.3. TCP Scalable updates its congestion window using fixed increase and decrease parameters (multiplicative increase and multiplicative decrease approach).

Two main parameters where introduced:

- $aiFactor$: Additive increase factor (50)
- $mdFactor$: Multiplicative decrease factor (0.125)

$TcpScalable$ inherits from $TcpNewReno$ the $SlowStart()$ method and proposes a new $CongestionAvoidance()$ and $GetSsThresh()$ method for congestion avoidance and fast retransmit respectively.

During Slow-start phase, the congestion window is increased by one packet for each acknowledgment received. In Congestion Avoidance, the window responds to each acknowledgment received with the update (constant parameter, $aiFactor$). In Congestion avoidance, the $cwnd$ grows by:

$INC = SMSS \times \left(\frac{ackCnt}{\min\{segCwnd\ ,aiFactor\}}\right)$, where $segCwnd$ is the number of segments in the current $cwnd$, $ackCnt$ is the Number of received ACK

In the event of congestion, each packet loss decreases the congestion window by a small fraction until packet loss stops.

$$ssThresh = SMSS\ \times \max\left\{2\ , (1 - mdFactor)\,\frac{inFlights}{SMSS}\right\}$$

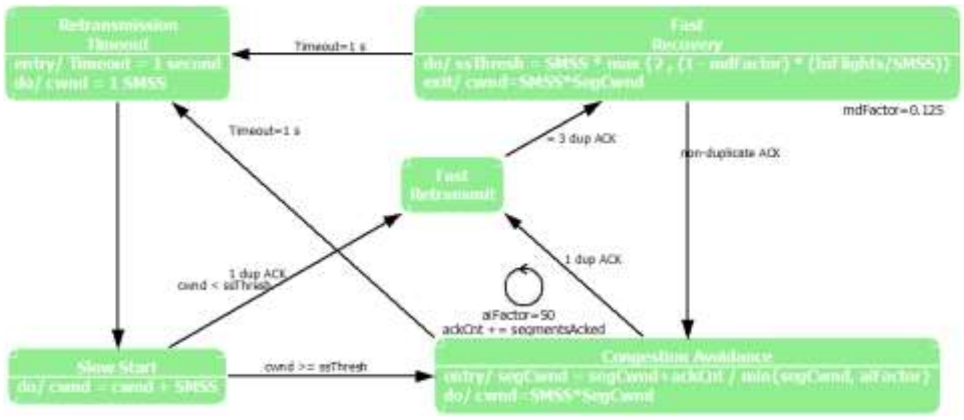


Figure 4.3.3: TCP Scalable's states diagram in ns-3.

4.3.4. TCP Highspeed

Designed for TCP connections with large congestion windows. This algorithm is in between Scalable and NewReno. TCP HighSpeed acts similarly to TCP NewReno when its congestion window size is less than the parameter $cwnd$ segments are less than 38. In any other way the Congestion Avoidance phase uses a table A to increment the $cwnd$ and table B to decrease $cwnd$.

$$INC = \frac{\mathrm{TableLookupA}\left(\frac{inFlight}{SMSS}\right)}{\mathrm{cwnd}}$$

$$DEC = 1 - \mathrm{TableLookupB}\left(\frac{inFlight}{SMSS}\right)$$

Where the tables A and B are based on the number of segments acknowledged.

The Figure 4.3.4 shows HighSpeed states diagram.

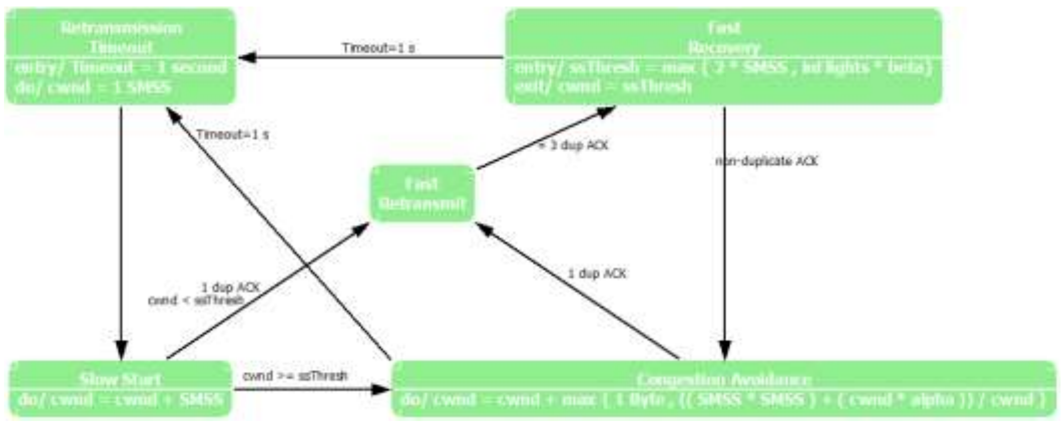


Figure 4.3.4: TCP Highspeed's states diagram in ns-3.

4.3.5. TCP Cubic

Cubic [16] is an enhanced version of BIC, which simplifies the BIC window control using a cubic function (which contains both concave and convex portions) and improves its TCP friendliness and RTT fairness. A key feature of Cubic is that its window growth depends only on the time between two consecutive congestion events and this time factor is known as congestion epoch.

The congestion window of Cubic is determined by the following function:

$cwnd = C(t-K)^3 + cwnd_{max}$ where: $K = \sqrt[3]{((\beta \times cwnd_{max})/C)}$

Where C is a cubic scaling factor, t is the elapsed time from the last window reduction, $cwnd_{max}$ is the window size just before the last window reduction, K is the time period that the function takes to increase $cwnd$ to $cwnd_{max}$ when there is no further loss event and β is a constant multiplication decrease factor applied for window reduction at packet loss or disorder event.

In Cubic, the methods $NewAck()$ and $DupAck()$ are inherited from $TcpSocketBase$. $NewAck()$ method is called for each new acknowledgment received and $DupAck()$ is called whenever a duplicate acknowledgment is detected. $DupAck()$ records the number of duplicate acknowledgments received to properly handle fast retransmit or for reducing $cwnd$. In addition to the inherited methods, $TcpCubic$ has added the following methods:

- The $Update()$ method that calculates cnt, which represents the ACK packets that should be received before increasing the $cwnd$ by one segment.
- The $CubicReset()$ method which resets CUBIC variables during timeouts.

Cubic class also implements a new slow start algorithm called $HyStart$ [17]. that reduces burst packet losses during slow start and hence achieves better throughput and a lower system load. The algorithm does not change the doubling of $cwnd$ during slow start phase but based on clues from ACKs pacing and round-trip delays, it heuristically finds safe exit points at which it can finish slow start and move to congestion avoidance before $cwnd$ overshoots. When packet losses occur during slow start, $HyStart$ class behaves in the same way as the original slow start algorithm.

$HyStart$ class is a plugin to the sender side of TCP and is easy to implement as it uses only TCP state variables available in common TCP stacks, [17].

The cubic state diagram and the congestion avoidance phases are depicted in Figure 4.3.5 and Figure 4.3.6 respectively. It shows that at the time of experiencing congestion event, the window size for that instant will be recorded as $cwnd_{max}$, which will be set as the inflection point of the cubic function that will govern the growth of the congestion window. After the lost event, the algorithm performs a multiplicative decrease of congestion window by a factor of β where β is a window decrease constant and the regular fast recovery and retransmit of TCP ($\beta = 0.8$). The transmission will then be set with a smaller window value and, if no congestion is experienced, the value of the window will continue to increase discreetly. 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.

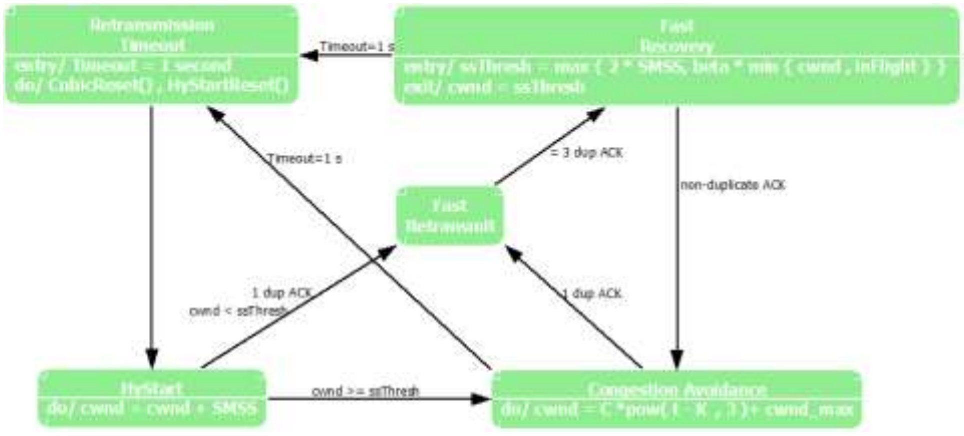


Figure 4.3.5: TCP Cubic's states diagram in ns-3.

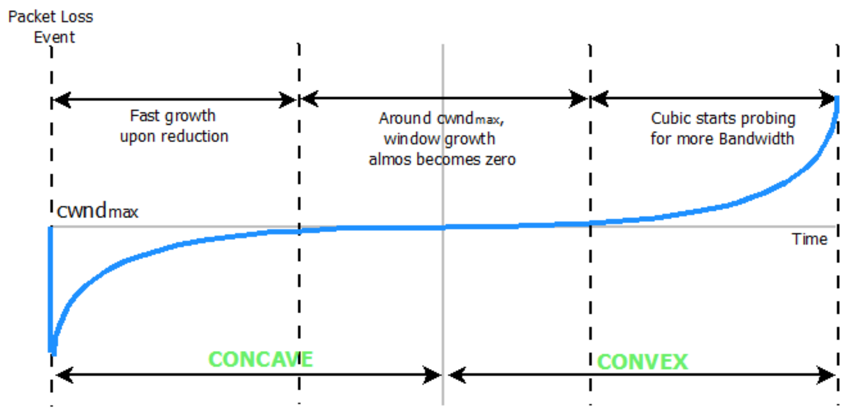


Figure 4.3.6: Window growth functions of CUBIC-TCP during congestion avoidance.

4.3.6. TCP HTCP

H-TCP (Hamilton TCP) is a congestion control protocol suitable for high bandwidth-delay product networks. This CCA strategy uses two functions for congestion avoidance (like $\alpha \times cwnd$) and fast retransmit phase ($\beta \times cwnd$), back off process.

HTCP inherits from $TcpNewReno$ the $SlowStart()$ method and proposes a new $CongestionAvoidance()$, $GetSsThresh()$ and $PktsAcked()$ methods for congestion avoidance, fast retransmit and track of the RTT_{min} and RTT_{max}respectively. Besides three new attributes are defined:

- $DefaultBackoff$: The default AIMD backoff factor (0.5).
- $ThroughputRatio$: Threshold value for updating beta (0.2).
- $Delta\_L$: increase function (1 seconds).

During congestion avoidance, HTCP considers the elapsed time since last congestion to calculate the factor α as a function of last congestion time ($Delta$), [25]:

$$\alpha(Delta) = \begin{cases} 1 & , Delta < Delta\_L \\ (1 + 10 \times (Delta - Delta\_L) + 0.25 \times (Delta - Delta\_L)^2), & otherwise. \end{cases}$$

$$\alpha = 2 \times (1 - \beta) \times \alpha(Delta) \rightarrow if(\alpha < 1): \ \alpha = 1$$

The value of α starts larger, if congestion has not occurred. This value increases the size of the congestion window every time an ACK is received.

The backoff coefficient (β) is calculated based on RTT_{max} and RTT_{min} since the last congestion and throughput ($Throughput_i$, maximum achieved throughput during the last congestion epoch) of the flow from the following relation:

$$\beta(i) = \begin{cases} 0.5 & , \frac{Throughput_i}{Throughput_{i-1}} - 1 \leq ThroughputRatio \\ \frac{RTT_{min}}{RTT_{max}} & , \qquad otherwise. \end{cases}$$

In congestion avoidance $cwnd$ grows as:

$$cwnd = cwnd + \max\{1, (SMSS^2 + cwnd \times \alpha)/cwnd\} \ Bytes$$

In recovery phase the $ssThresh$ is reduced as:

$$ssThresh = \max\{2 \times SMSS \, , inFlight \times \beta\}$$

H-TCP aims at a faster convergence and better utilization by setting α to be an increasing function of the time elapsed since last backoff and setting β to be such that the link is always around full utilization, even after the back off.

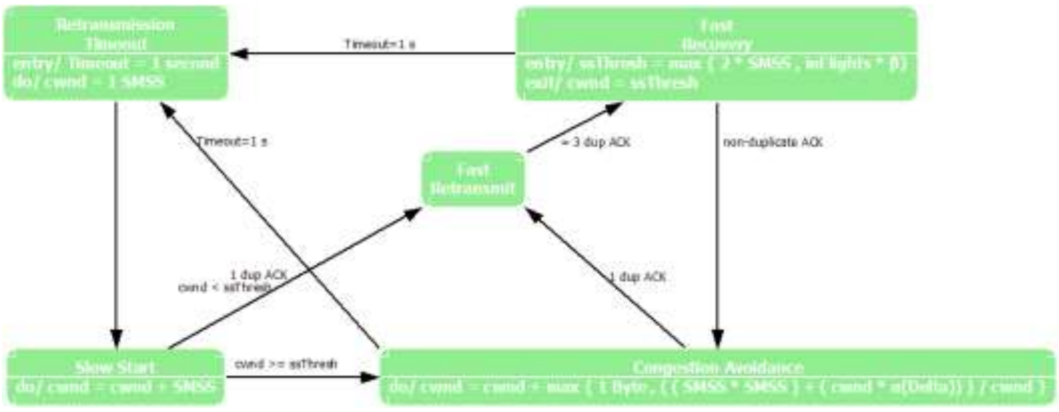


Figure 4.3.7: TCP HTCP's states diagram in ns-3.

4.3.7. TCP Illinois

TCP Illinois is a Hybrid-based congestion control algorithm, which uses two congestion indicators as feedback. Firstly, the packet loss to determine whether the window size should be increased or decreased and secondly the queuing delay to determine the amount of the increment/decrement, C-AIMD (concave-AIMD) approach in congestion avoidance.

Illinois retains the $SlowStart()$ method from $TcpNewReno$ and proposes a new $CongestionAvoidance()$, $GetSsThresh()$ and $PktsAcked()$ methods for congestion avoidance, fast retransmit and track of RTT_{min} ,RTT_{max} , the summation of all RTT measurements during last RTT ($sumRtt$) and the number of RTT measurements during last RTT ($cntRtt$).

Moreover, four new main attributes were defined:

- $AlphaBase$: Alpha base threshold (1.0), bounded by $[AlphaMin(0.3); AlphaMax(10)]$.
- $BetaBase$: Beta base threshold (0.5), bounded by $[BetaMin(0.125); BetaMax(0.5)]$.
- $WinThresh$: Window threshold (15).
- $Theta$: Theta threshold (5).

TCP Illinois proposes two functions models, $\alpha(.)$ as the additive increase factor which is large when far from congestion and a small value when close to congestion:

$$\alpha_{(da,dm)} = \begin{cases} AlphaMax & , \; da \le d1 = \frac{dm}{100} \\ \frac{dm \times AlphaMax}{dm + \frac{da \times (AlphaMax - AlphaMin)}{AlphaMin}} & , \; da > d1 \end{cases}$$

Where: $dm = RTT_{max} - RTT_{min}$ refers to the maximum average queuing delay and $da = \frac{sumRtt}{cntRtt} - RTT_{min}$ is the current average queuing delay which stays below $d1$ for some $theta$ amount of time.

The function β represents the multiplicative decrease factor which should be small when far from congestion and large when close to congestion in order to achieve a better throughput in case the packet loss is not in fact caused by congestion events.

$$\beta_{(da,dm)} = \begin{cases} BetaMin & , \; da \le d2 = \frac{dm}{10} \\ \frac{BetaMin \times d3 - BetaMax \times d2 + (BetaMax - BetaMin) \times da}{d3 - d2} & , \; d2 < da < d3 = \frac{8 \times dm}{10} \\ BetaMax & , \; da \ge d3 \end{cases}$$

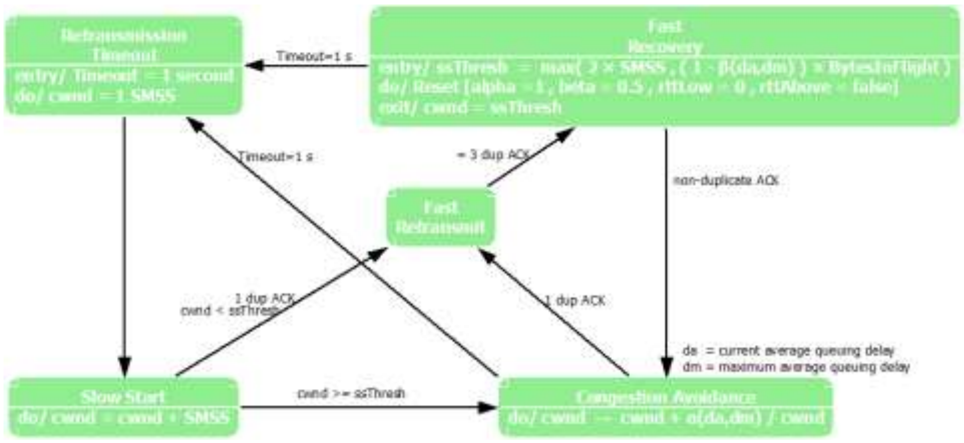
In the congestion avoidance phase, the sender measures RTT for each ACK and average the RTT measurements over the last $cwnd$ acknowledgements (one RTT interval) to average the RTT. The sender maps the maximum and minimum RTT ever seen and computes the maximum average queueing delay dm and the current average queueing delay da.

If the congestion window is below $WinThresh$ the sender sets the parameters to $\alpha = 1$ and $\beta = 0.5$. The sender computes α and β values once per RTT as a result, when the average queuing delay is small the sender assumes no risk of congestion event and sets a large α and small β and when the average queuing delay is large the sender estimates an imminent congestion event and sets a small α and large β. In congestion avoidance $cwnd$ grows as proportional as $cwnd \rightarrow cwnd + \alpha/cwnd$.

If in the last RTT there is packet loss detected through triple duplicate ACK, the slow start threshold is calculated as:

$$ssThresh = \max(2\,SMSS, (1 - \beta) \times BytesInFlight)$$

TCP-Illinois increases the throughput much more quickly than TCP when congestion is far and increases the throughput very slowly when congestion is imminent. As a result, the window curve is a concave curve and the average throughput achieved is much larger than the standard TCP


Retransmission Timeout
entry/ Timeout = 1 second
do/ cwnd = 1 SMSS
Timeout=1 s
Fast Recovery
entry/ ssThresh = max(2 × SMSS , (1 - β(da,dm)) × BytesInFlight)
do/ Reset [alpha = 1 , beta = 0.5 , rttLow = 0 , rttAbove = false]
exit/ cwnd = ssThresh
Timeout=1 s
= 3 dup ACK
non-duplicate ACK
Fast Retransmit
1 dup ACK
cwnd < ssThresh
1 dup ACK
da = current average queuing delay
dm = maximum average queuing delay
Slow Start
do/ cwnd = cwnd + SMSS
cwnd >= ssThresh
Congestion Avoidance
do/ cwnd ← cwnd + α(da,dm) / cwnd
Figure 4.3.8: TCP Illinois's states diagram in ns-3.

5. TCP Performance Analysis for Dual Connectivity

This chapter presents and discuss the simulation results of the TCP protocols discussed in the previous, chapter 4. The simulator modelled a macro cell (LTE eNB) with 3 small sell (5G nodes) scenario. The small cells are configured to use the 28 GHZ frequency band. The UE configured as a client communicating with a server (Remote Host), while it moves within the network. The small cells are mmWave gNBs that are managed by the LTE eNB (coordinator).

Different TCP congestion control protocols were implemented. The congestion algorithms executed are NewReno, Hybla, Scalable, HighSpeed, Cubic, Illinois and HTCP. In addition, each algorithm is tested at 1000Mbps and 3000Mbps application rate. In addition, we played with the values of the radio link control buffer and the X2 latency in order to analyse the relevant trade-offs in the system.

Each TCP congestion control algorithm is analysed based on the congestion window, Round trip time, Bytes in-flight and the application layer goodput.

Figure 5.1.1 describes the user equipment's mobility scenario proposed, where the user plane is provided by the SeNB node and the control plane by the MeNB or LTE eNB. The LTE eNB provides a backup link for the user plane in cases of outage. In this scenario the UE goes across the scenario while it performs Fast switching and secondary cell handover in a TCP session. The subsection 5.1 gives more details of the test study in this work.

5.1. Simulation Scenario

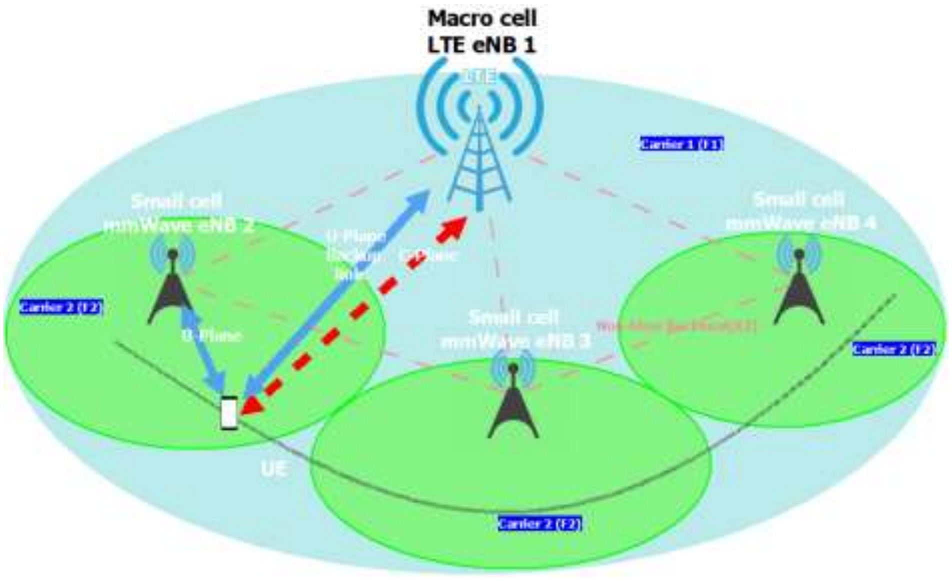


Figure 5.1.1: Radio access topology.

Due to the complexity of the scenario proposed, the test campaign is split into 2 phases. The first phase designs the scenario with secondary cell handovers without blockages (no buildings), and the second scenario allocates buildings between the mmWave gNBs.

The scenario 1 is shown in Figure 5.1.2, there is an LTE eNB node placed at point x=205 and y= 100, at a height of 30 meters, and three mmWave gNB nodes which are allocated along the y-axis at 35 and separated by 200 meters and each of them is 10 meters high. The scenario 2 is shown in Figure 5.1.3, where the buildings were placed between the mmWave gNBs. Both scenarios used the parameters in Table 5.1, where the user is set with an antenna 8x8 uniform planar array and the mmWave gNB with a set of 16x16 uniform planar array, the hysteresis for switch handover is set to -5 dB and for secondary cell handover is set to 3 dB.

The user moves along x-axis at speed of 5 m/s ($speed_{UE} = 5m/s$) from coordinate (0, -5) to (415, -5) ($Distance = 415\ meters$). Therefore, the simulation duration ($Simulation_{time}$) that depends on the user speed is the total path divided by the UE speed, 83 seconds.

$$Simulation_{time} = \frac{Distance}{speed_{UE}}$$

All traffic across the network are generated using Application class with an MTU size of 1400 bytes, Table 5.2 shows the TCP parameters configured for the simulations such as the socket ($TcpSocketFactory$) type used, timeout parameter at 1 second and the initial slow start threshold at 32 Mbytes. The Remote host has a point-to-point connection to the PGW/SGW router with bandwidth of 1000 Gbps with delay of 10 milliseconds. The logical link between the PGW/SGW router and the LTE eNB has a bandwidth of 10 Gbps with a delay of 1 millisecond. The X2 is a point-to-point link with bandwidth of 1000 Gbps with the bottleneck delay ranges from 0.5 milliseconds to 2 milliseconds to cover the different delays for various network environments. The millimeter wave radio access is implemented with the $3ggpchannel$ class, Table 5.1. The $3gppchannel$ classes implement different features like pathloss model, beam management, blockage, etc.

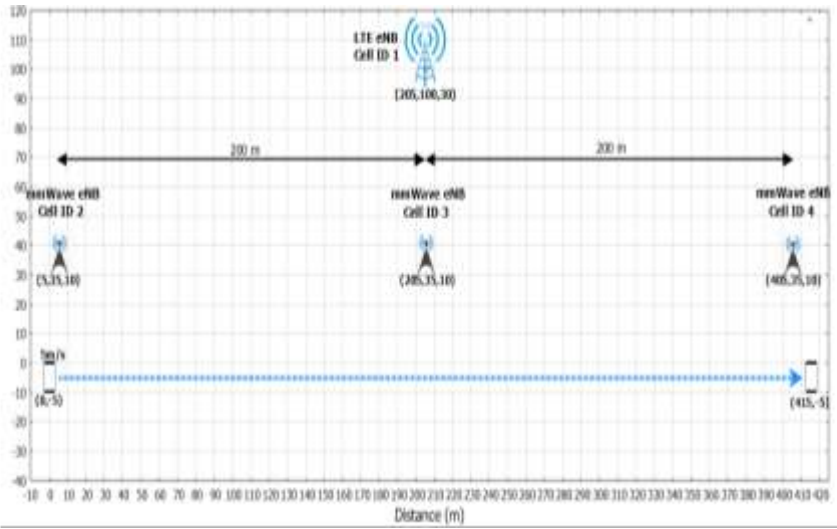


Figure 5.1.2: UE mobility scenario 1

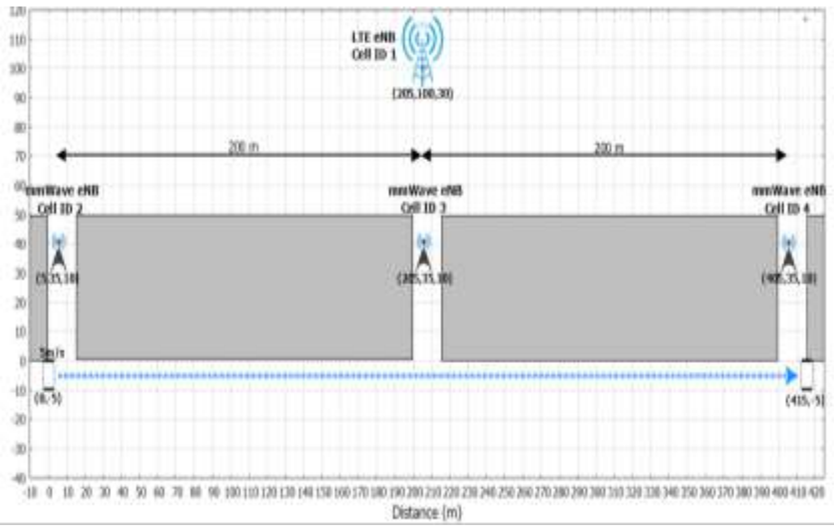


Figure 5.1.3: UE mobility scenario 2

Parameter	Value	Description
Small cell BW	1 GHz	Bandwidth of mmWave gNBs
Small cell fc	28 GHz	mmWave carrier frequency
Small cell UL-DL Ptx	30 dBm	mmWave transmission power
Macro cell BW	20 MHz	Bandwidth of LTE eNBs
Macro cell fc	2.1 GHz	LTE downlink carrier frequency
Macro cell DL Ptx	30 dBm	LTE transmission power for downlink
Macro cell UL Ptx	23 dBm	LTE transmission power for uplink
Number of macro cells	1	LTE eNB
Number of small cells	3	mmWave gNB
Number of UE	1	Dual connectivity user
Small cell antenna	16x16 UPA	mmWave gNB antenna
UE antenna	8x8 UPA	mmWave UE antenna
ISD	200 m	Inter-Site Distance
F	5 dB	noise Figure
▲LTE	-5 dB	Threshold for switch/handover to LTE
▲hys	3 dB	Hysteresis for handover
Path Loss model	MmWave3gppBuildingsPropagationLossModel	UMi-StreetCanyon
Channel model	MmWave3gppChannel	3GPPChannelModel

Table 5.1: Scenario parameters

Traffic type	TCP ($TcpSocketFactory$)
MTU	1500B
TCP socket type variant	Cubic, Hybla, TcpNewReno, Highspeed, Htcp, Scalable and Illinois
Packet size	1400B
MSS	1400B
Initial slow start threshold	32000000 Bytes
Timeout	1s
Simulation Tool	ns-3
Performance metric	Goodput RTT PDR Bytes in flight

Table 5.2: TCP Simulation parameters

5.2. Simulation parameters and data collection

Different combinations such as RLC buffer size, X2 latency, TCP protocol were defined to test the performance of single user (client) connected to the Remote Host (server).

The RLC buffer sizes used for simulations were defined considering the product of round-trip time (RTT) and the throughput target at 25 milliseconds and 1Gbps respectively, which later we will see that it is linked with the network in flight bytes. In addition, the implemented queue manager at the RLC layer is the Drop-tail algorithm, configuring a maximum queue length to 1M packets.

The combination of parameters, e.g. X2 latency, RLC buffer and type of protocol, for the simulation were repeated 10 times per set of combination and averaged. The simulations performed are shown in Table 5.3. The main metrics analysed are the goodput, RTT, Bytes in flight with deviation and simulations accuracy.

	Scenario 1		**Scenario 2**	
	Test A	**Test B**	**Test C**	**Test D**
Target Application rate	1000 Mbps	3000 Mbps	1000 Mbps	3000 Mbps
X2 latency D_{X2}	500, 1000 and 2000 microseconds	500, 1000 and 2000 microseconds	500, 1000 and 2000 microseconds	500, 1000 and 2000 microseconds
RLC Buffer size B_{RLC}	1 MB, 5 MB and 10 MB	5 MB, 10 MB and 20 MB	5 MB, 10 MB	5 MB, 10 MB
TCP Variants	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois
Number of simulations	10	10	10	10
Simulation time	1.5 hours approx.	2.5 hours approx.	1 hours approx.	2 hours approx.
Total number of simulations	3x3x7x10=630 simulations	3x3x7x10=630 simulations	3x2x7x10=420 simulations	3x2x7x10=420 simulations
Total time	945 hours	1575 hours	420 hours	840 hours

Table 5.3: Simulation variables and number of simulations, Test A and C (1000 Mbps) and Test B and D (3000 Mbps).

5.3. TCP Performance Insights per CCA

In this section, the simulation outcomes of the different TCP variants described in chapter 4 are presented. In order to explain the performance of the congestion control algorithms one set of parameters were chosen to describe the protocol performance.

The TCP options enabled for a single TCP flow are selective acknowledgement (SACK) and window scale to achieve high values of congestion window. In addition, the time stamp option is also enabled to achieve more accurate round-trip time (RTT) measurements at Transport layer.

The RLC layers were configured with buffer values around the BDP, additionally, the TCP nodes have enough space for packets at the application layer, to transmit data for the full duration of the simulations.

For this analysis, the values of X2 latency and RLC size were set to 500 microseconds and 10 MBytes respectivetly. Meanwhile the target rate and scenario type variated as depited in Table 5.4.

The CCA were set up to use SACK, this means that in recovery phase the $cwnd$ is set to $ssThresh + (dupAckCount \times SMSS)$.

	Scenario 1		Scenario 2	
	Test A	**Test B**	**Test C**	**Test D**
Target Application rate	1000 Mbps	3000 Mbps	1000 Mbps	3000 Mbps
X2 latency $\boldsymbol{D_{X2}}$	500 microseconds	500 microseconds	500 microseconds	500 microseconds
RLC Buffer size $\boldsymbol{B_{RLC}}$	10 MB	10 MB	10 MB	10 MB
TCP Variants	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois	Cubic, Hybla, NewReno, Highspeed, HTCP, Scalable and Illinois

Table 5.4: Simulation variables for ***Error! Reference source not found.*** *section* ***Error! Reference source not found.*.**

5.3.1. Congestion Events

In this section, the cases where the sender detects packet loss events are exemplified. Figure 5.3.1 shows NewReno variant used as a reference to identify in which cases the TCP sender interprets the arrival of duplicate acknowledgements or not arrival of ACKs as a sign of congestion in the network.

As we can see in Figure 5.3.1, there are four cases:

- During slow start phase, subsection 5.4.1.
- Due to Blockage (buildings), in following subsection the test B and D refer to this effect.
- During Handovers (Fast switching and secondary cell handover), subsection 5.4.3.
- Not dominance (for urban scenarios, in the region that equidistant between mmWave nodes), showed in Figure 5.3.2(c).

For such packet loss/congestion cases, NewReno protocol increases the congestion window size by one packet whenever a whole congestion window worth of data has been acknowledged in slow start. However, if the receiver obtains a packet that has a sequence number higher than the expected one, a duplicate ACK is sent by the receiver. In this way the sender is informed that the packet with the expected sequence number has been lost and requires retransmission. If three duplicate ACKs are received the sender enters in fast retransmit mode by setting the congestion to the new slow start threshold, $\max\left\{2\,SMSS,\ \frac{inFlights}{2}\right\}$, which in this case is half of Bytes in-flight. Then, the sender returns to congestion avoidance phase after it finished retransmitting the missing packets.

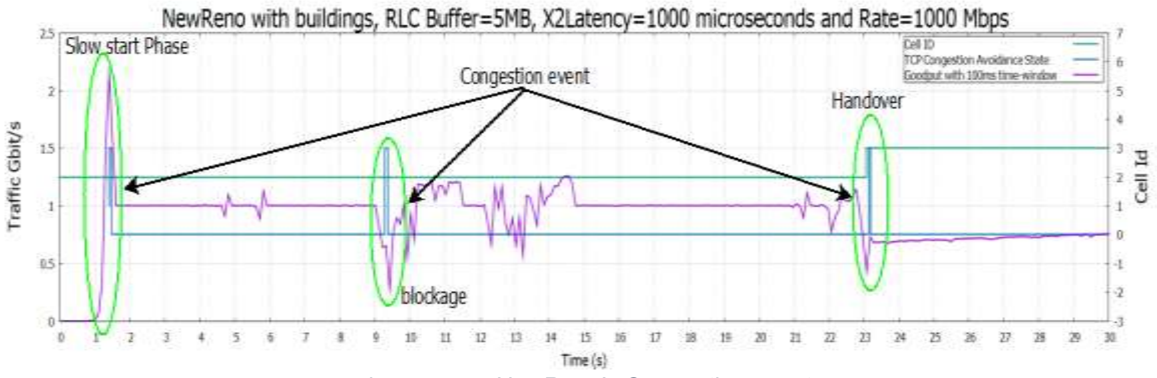


Figure 5.3.1: NewReno's Congestion events

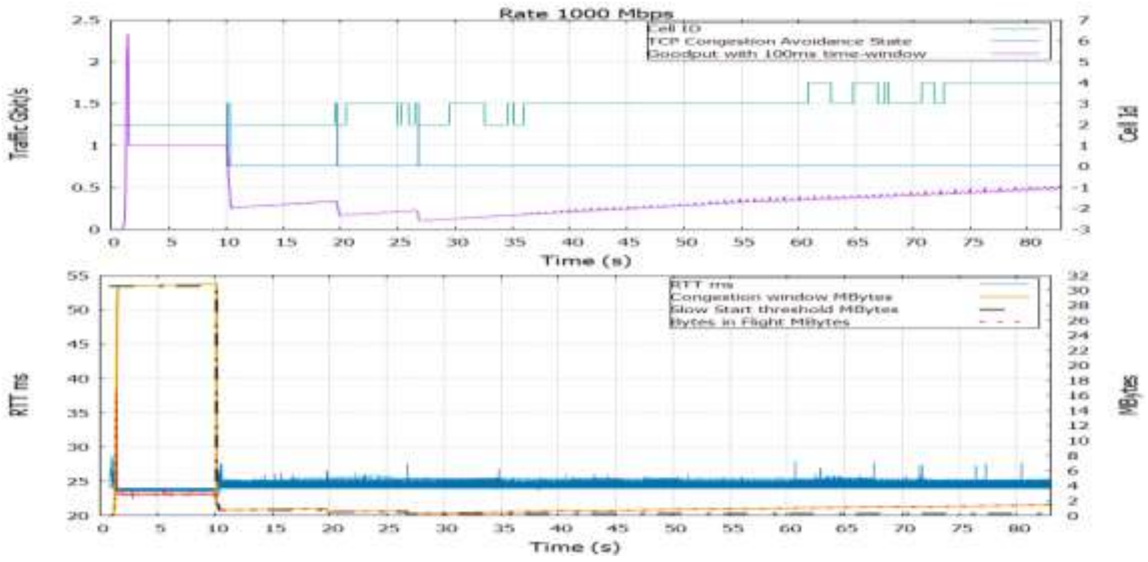
The Figure 5.3.1 shows the first 30 seconds of the simulation with a time window of 100 milliseconds for scenario 2. As we can see the protocol reached the target rate, however as we vary the X2 latency and RLC buffer the performance will dramatically change. The TCP congestion states goes between 0 (OPEN),1 (DISORDER), 3 (RECOVERY) and 4 (LOSS), described in section 4.2.3.

In the next sections, these events are analysed for different congestion algorithm.

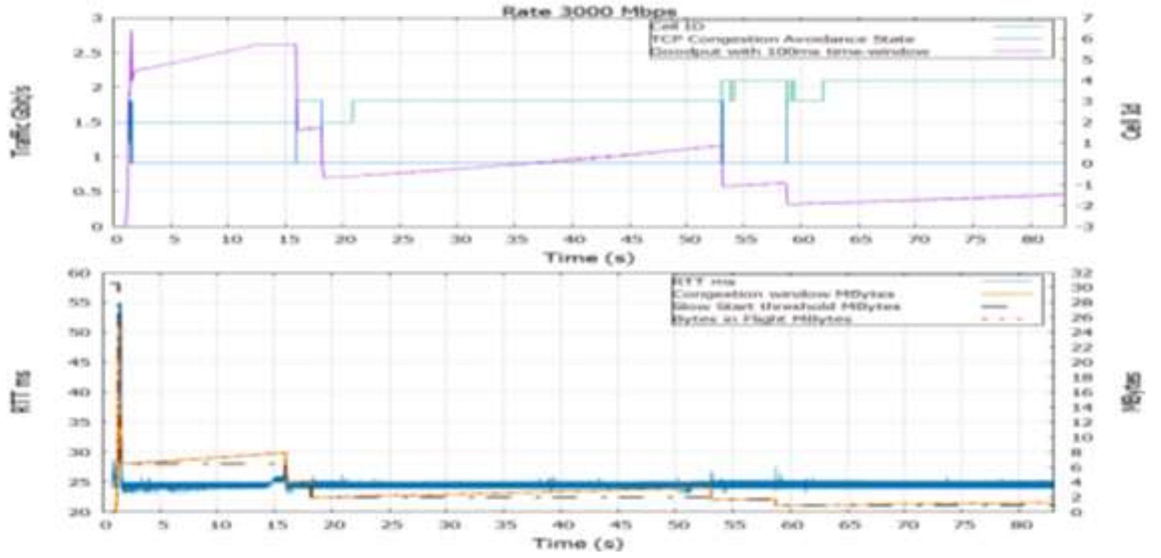
5.3.2. NewReno

This experiment is carried out in order to have a better understanding of the influences of NewReno variant on the performance of TCP over mmWave. The results are presented in Figure 5.3.2.

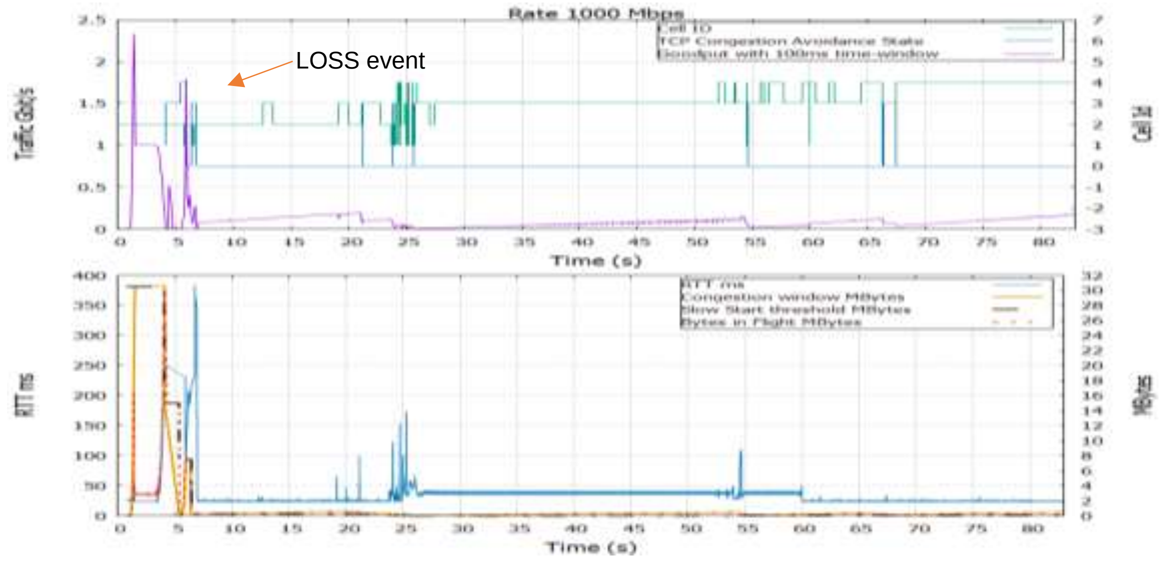
In scenario 1, the test A quickly achieves the initial slow start threshold to move congestion avoidance while holds the speed target for different configurations of B_{RLC} and D_{X2}, but only for the first 10 seconds. It is interrupted by the first congestion event (CA_DISORDER), caused by handovers, and the receiver starts sending dup ACK to the sender. After the third dup ACK, the sender launches the recovery mode, where the congestion window and slow start threshold are reduced to halve of the number of Bytes in-flight and the sender retransmits the missing unacknowledged packets. On the other hand, test B experiments congestion during the slow start phase, the system overflows at the RLC (we can see in Figure 5.3.2 that occupancy of Bytes in-flight goes to 26 Mbytes instantaneously, along the system elements), and moves to congestion avoidance before it achieves the slow start threshold value, congestion event during slow start phase. The slow start turns very aggressive at the end overflowing the network.



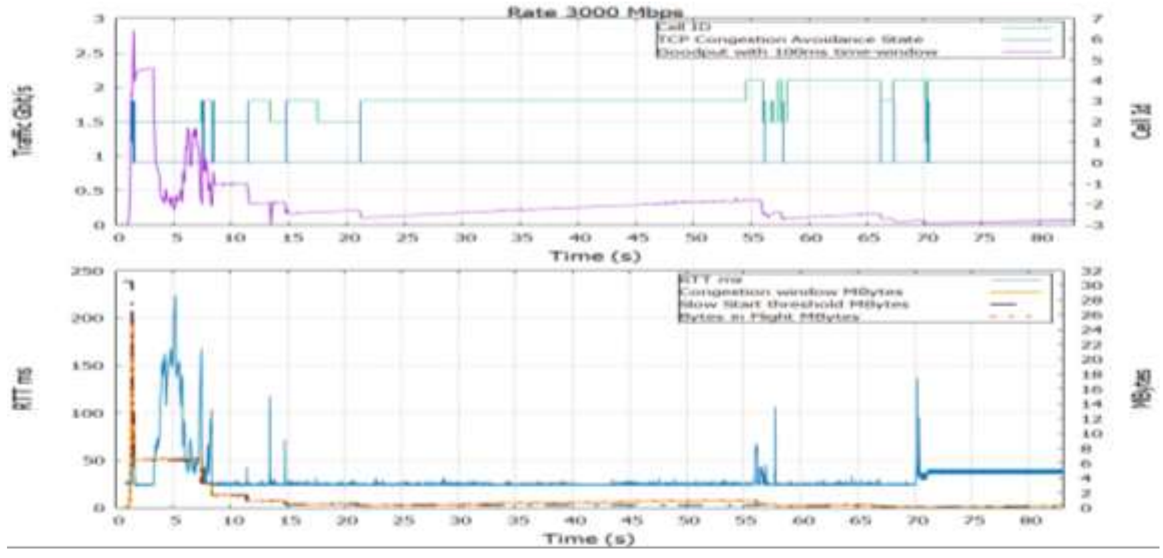
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.2: TCP NewReno performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

For test C, in non-line of sight area the channel conditions are degraded incrementing the erroneous decode of signal, as we can see from Figure 5.3.2. The SINR level went to 0 dB (the red and green dots in Figure 5.3.3) so the system performed a Fast switching to the LTE eNB to maintain the transmission. In terms of speed the user moves from 1 Gbps (mmWave gNB) to 40 Mbps (LTE eNB) generating delay in the communication despite the transition to LTE was very short the time runs out and TCP understood that the packet was lost. In Test D, apart from the congestion generated by the slow start phase there is another issue. The modulation scheme for non-line of sight conditions is reduced so there is an extra delay in the transmission as we can see in Figure 5.3.2 (d) where in RTT increases up to 6 times its usual value. In this case the buffer is too large, it is reflected in the RTT spike between 5 to 15 seconds of the simulation.

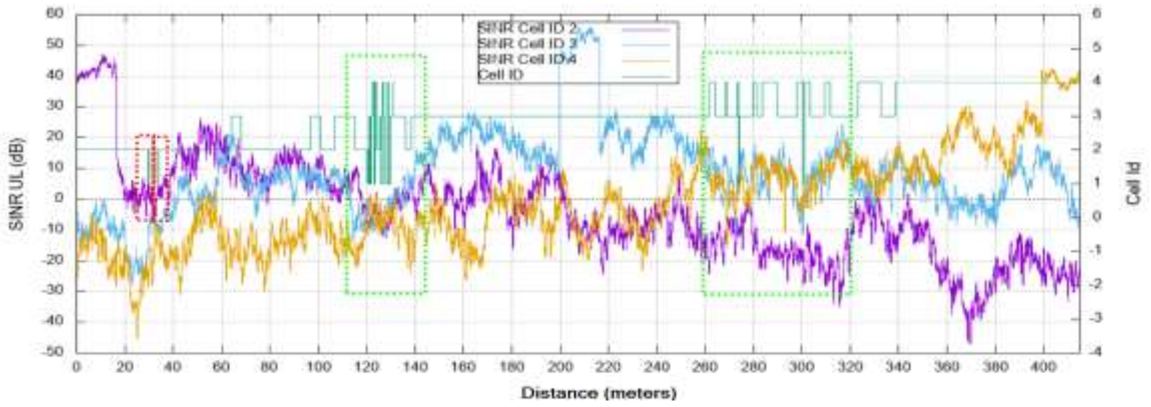


Figure 5.3.3: Test C, SINR value at the mmWave gNB. Congestion caused by building blockage (red dots) and Congestion by not dominance signal between mmWave gNBs (green dots).

Based on the results, New Reno poorly increases its throughput because the $cwnd$ is incremented by roughly 1 full-sized segment per round-trip time. This growth trend is very slow and makes almost impossible to achieve high rates. The system needs larger buffers to handle the number of Bytes in-flight during slow start and lower buffers for high latencies.

The Figure 5.3.4 and Figure 5.3.5 are showing the overall results of NewReno for scenarios 1 and 2. Figure 5.3.4 shows how the round-trip time is affected by the increment of the X2 Latency and traffic (due to increase of Bytes in-flight tenancy). Besides, the standard RTT deviation shows how much measurements are spread out from the average (mean) for each simulation, this is reflected in the scenario 2 where the RTT fluctuates because of the varying channel environments. Finally, the green line represents the average goodput, which is degraded when the system is set up with X2 latency. The best situation is when the system is set up with the lowest X2 latency (500 microseconds) in most of the cases but still below the rate targets.

In Figure 5.3.5, the congestion window shows a rise when we incremented the RLC buffer. For tests A and C there is a waste of resources at 10 Mbytes buffer capacity as for test B and D. The number of Bytes in-flight and the congestion window growth have the same trend to reach the maximum capacity but the congestion window AIMD strategy turns into a limitation for the system the performance.

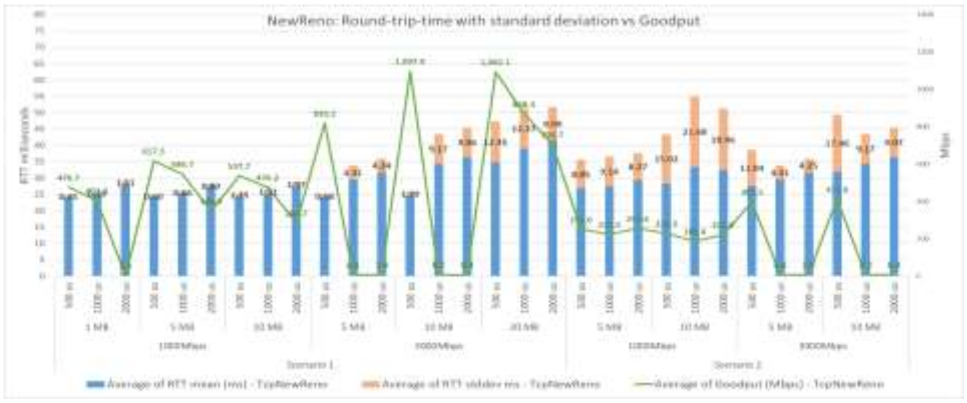


Figure 5.3.4: NewReno's Round trip time (RTT).

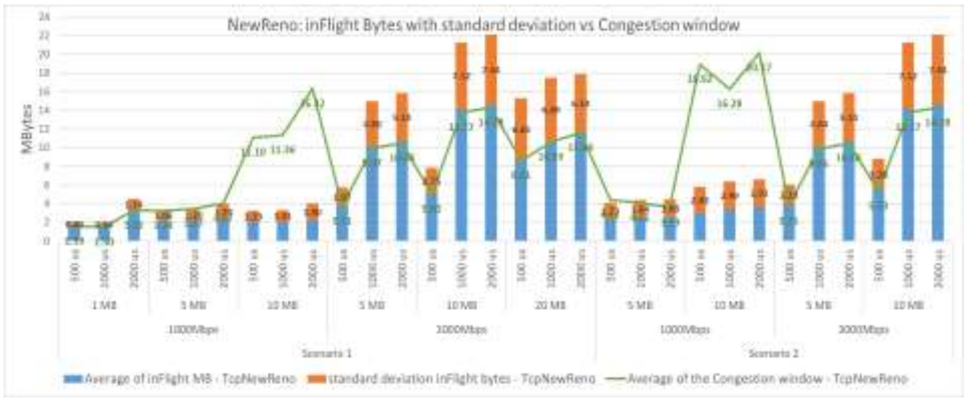
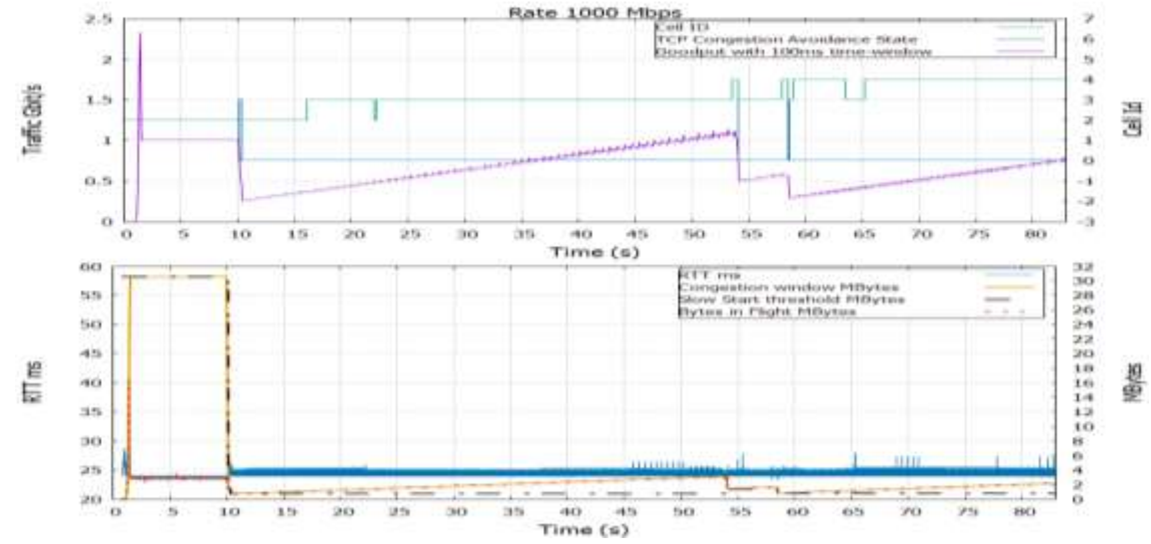


Figure 5.3.5: NewReno's inflight Byes (blue bars) vs Congestion Window (green line)

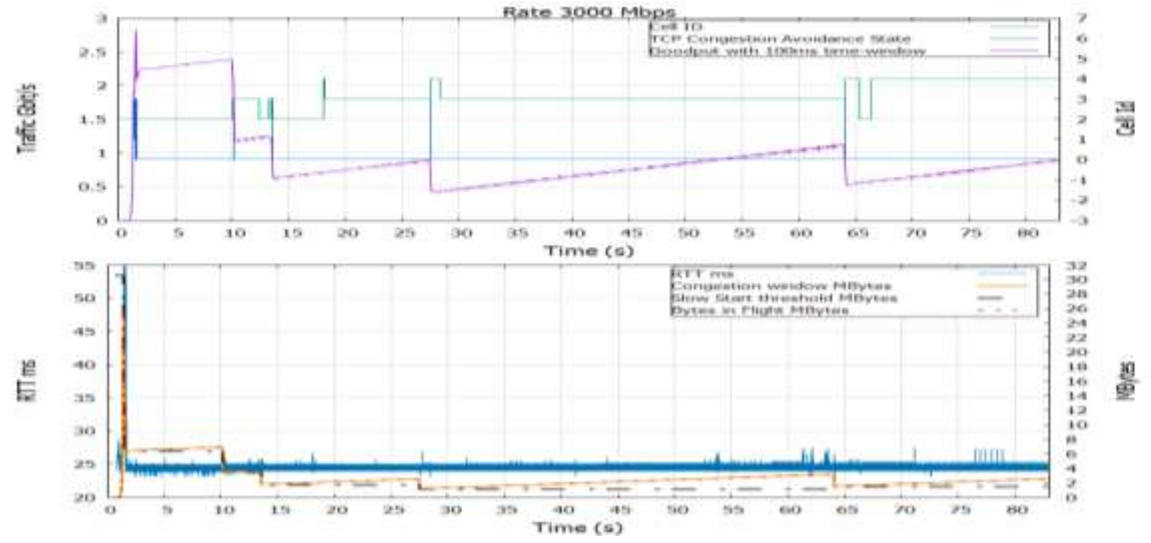
5.3.3. Hybla

The results of TCP Hybla over Dual Connectivity (DC) system are depicted in Figure 5.3.6, Figure 5.3.8 and Figure 5.3.9. For scenario 1, the results show a performance like that observed in New Reno. The only notable difference in this test is that during congestion avoidance the $cwnd$ increases steeply as compared to NewReno. In Test C, apart from the congestion event caused by NLOS conditions, after 15 seconds the bad channel condition for downlink transmission (0 MCS index). The occupancy of the Bytes in-flight in the system increment the latency which increment the probability of timeout events and the risk to fall in constant RTO case (Section 5.4.2). Figure 5.3.7 in sixth second, we can see how the signal strength goes under 0 dB, since the RLC layer has the capacity to buffer tis data in this case is a disadvantage to have a 10 Mbytes buffer. In Figure 5.3.8 we can see how reducing the buffer at 5 Mbytes the Goodput slightly improve.

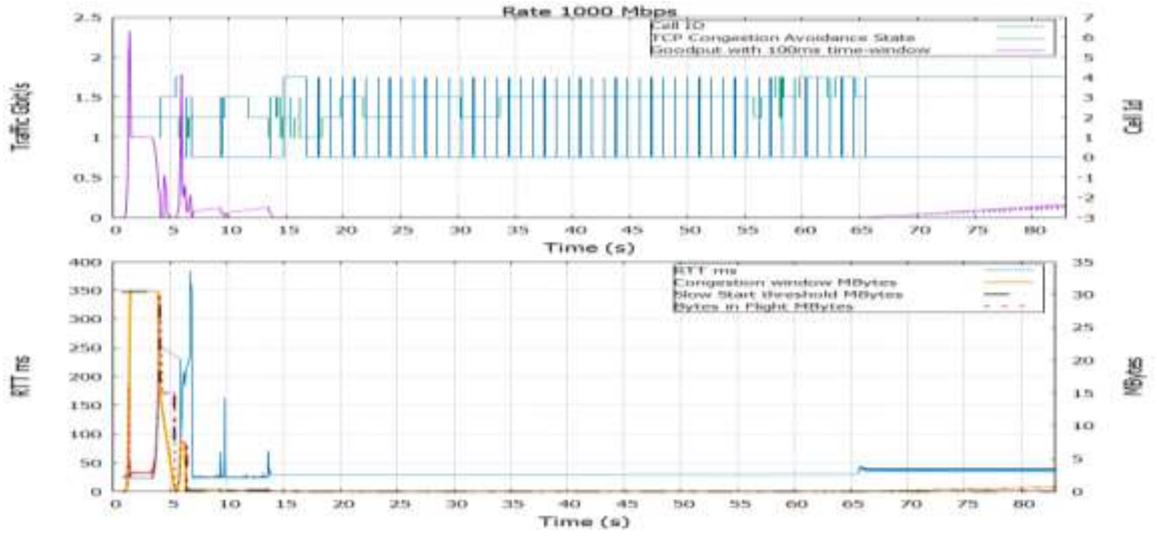
The Figure 5.3.6(d) shows a better performance of Hybla against NewReno thanks to its normalized rho parameter but due to the constant congestion window reduction (congestion events) and by byte increment strategy during congestion avoidance the performance is very poor after the first 15 seconds.



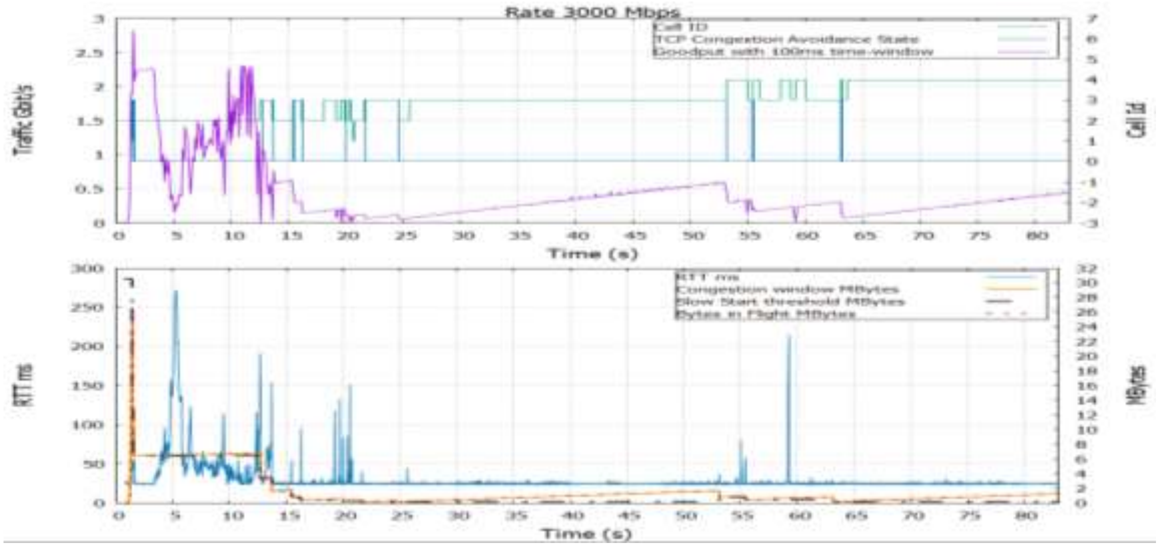
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.6: TCP Hybla performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

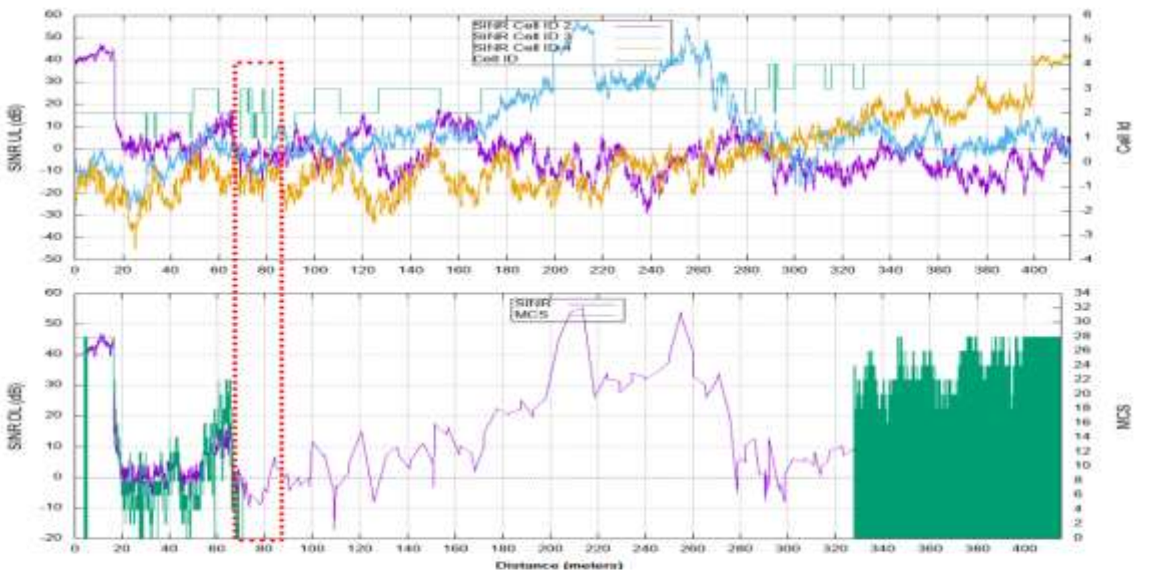


Figure 5.3.7: Test C Hybla's UL/DL SINR (dB) and MCS vs Distance (meters). The red dots show a LOSS congestion event caused by bad channel condition (MCS index 0).

The Figure 5.3.8 and Figure 5.3.9 are showing the overall results of Hybla for scenarios 1 and 2. From Figure 5.3.8, as we saw in NewReno the round-trip time is affected by the increment of the X2 Latency and traffic (due to augmentation of Bytes in-flight tenancy). Like NewReno the average goodput results are degraded when the system is set up with X2 latency. The best situation is when the system is set up with the lowest X2 latency (500 microseconds) in most of the cases but still below the application rate target (tests A and C have a target of 1000 Mbps and tests B and D have a target rate of 3000 Mbps, Table 5.3).

In Figure 5.3.9, the increment of the RLC buffer allows an increase in the congestion window size. Tests A and C show a waste of buffer resources at the 5 and 10 Mbytes. Tests B and D show that the number of Bytes in-flight and the increment of the congestion window size is constraint to the strategy during congestion avoidance.

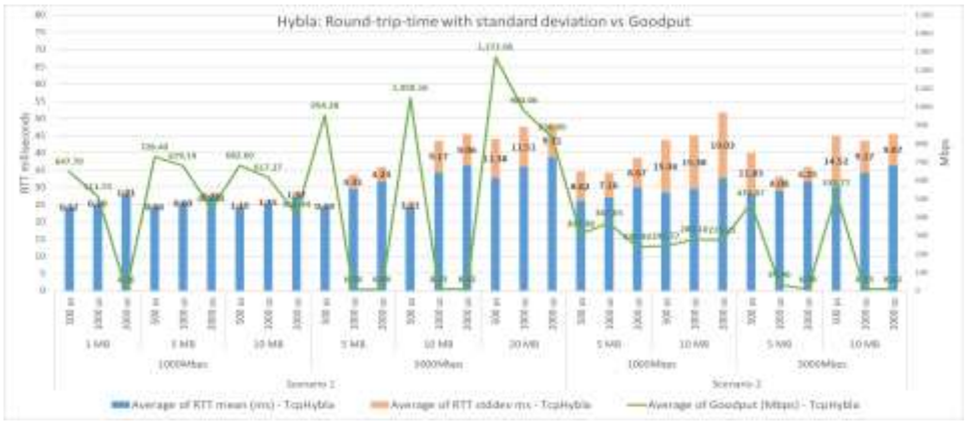


Figure 5.3.8: Hybla's Round trip time (RTT) and goodput.

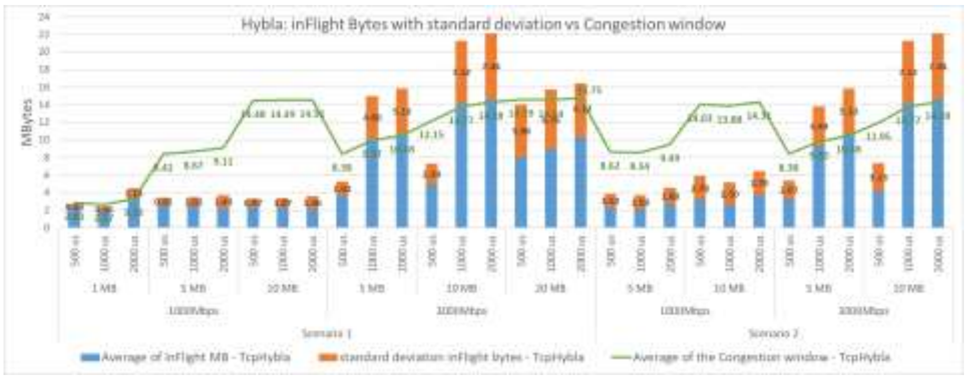
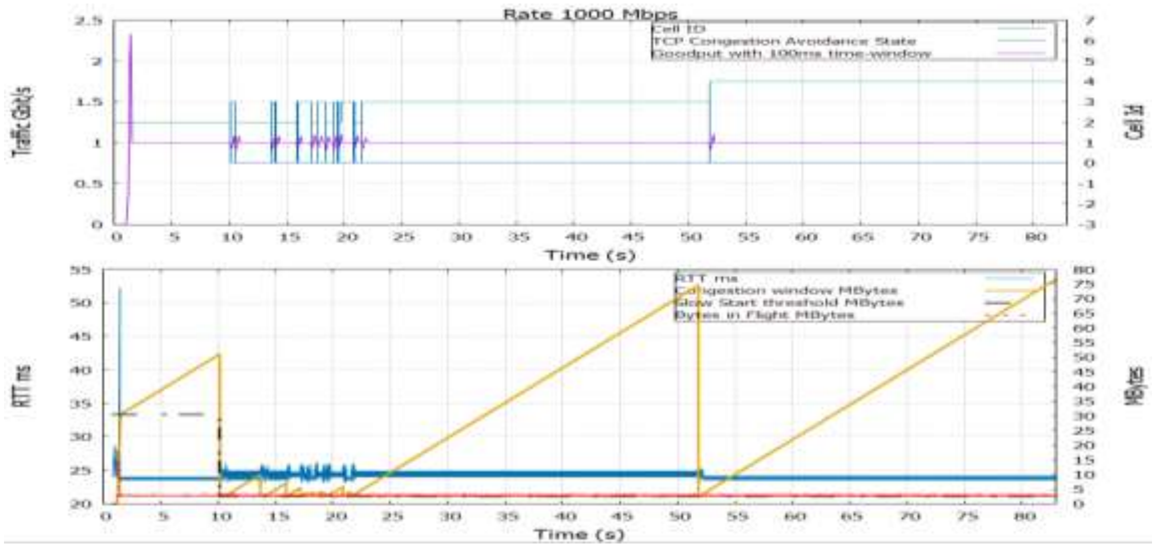


Figure 5.3.9: Hybla's inflight Byes (blue bars) vs Congestion Window (green line)

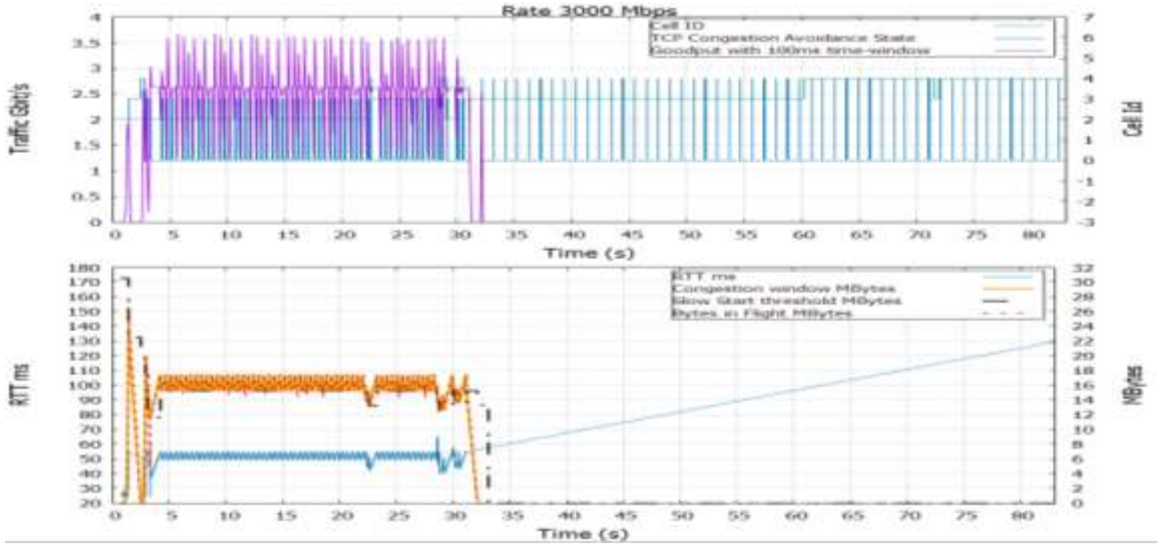
5.3.4. Scalable

In order to identify the impact of TCP Scalable on the performance of Dual Connectivity (DC) system, the model is implemented and Figure 5.3.10 present the results. The results show a different performance than the one observed with New Reno. After congestion events, the $cwnd$ is reduced and the algorithm aggressively increments the $cwnd$ (in section 4.3.3 the equation to calculate the INC increments proportionally to the number of segments acknowledged divided by the minimum value between the number of segment in the current window and 50, in this way the traffic quickly converge to the target application rate), so it fast returns to the target rate (1 Gbps), as we can see on the left side of Figure 5.3.10. But when the system is tested at its maximum capacity, the network constantly experiences CA_DISORDER events and the Bytes in-flight saturate the system. As a result, the RLC buffer overflows and it starts to drop packets, which causes packet loss events. The performance will deteriorate further as the user approaches to the next cell region and handover process moves the buffers through the X2 link, incrementing the latency. Due to the large buffers, there still are duplicate ACKs in flight received after the initial retransmission time out (RTO). These are causing the TCP state machine to go right back into fast retransmit after the timeout, even though it probably should continue slow start, the issue is also addressed to ns-3, the process is explained in section 5.4.2.

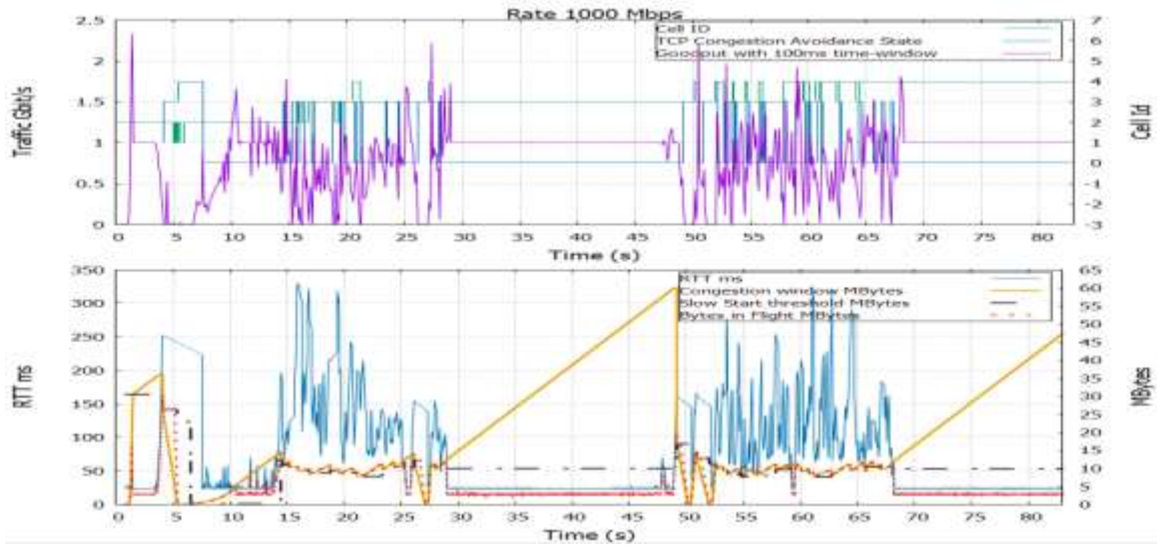
In Tests C and D, the delay caused by channel conditions with large buffer is evidenced because again the aggressive strategy instigating constant drops.



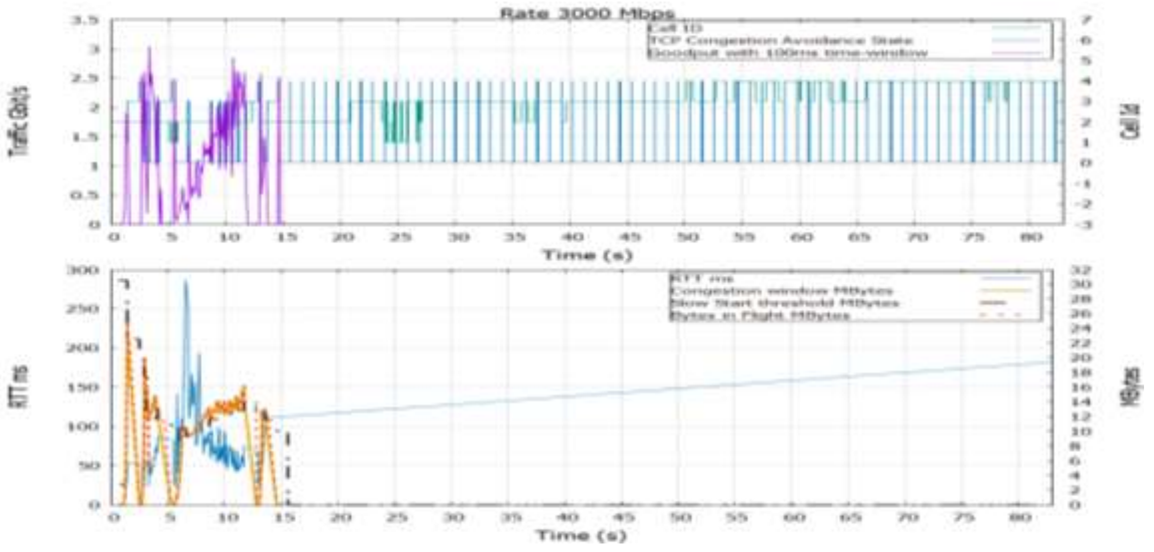
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.10: TCP Scalable performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

Figure 5.3.11 and Figure 5.3.12 show that for Test A the RTT performance is stable and the goodput is very close to the target rate, however as we saturate with more packets the network is constantly overflows the system. In scenario 2, TCP Scalable probes to overcome congestion window reduction and large RTT, but its $cwnd$ growth increases the probability RTO due to large latencies.

TCP Scalable is more efficient than NewReno, its efficiency is very high, but it causes collateral damage to the element in the network (buffer overflow). As a result, Scalable congestion control strategy is too violent for the network, the congestion window grows too large, so the system overflows and packets are dropped.

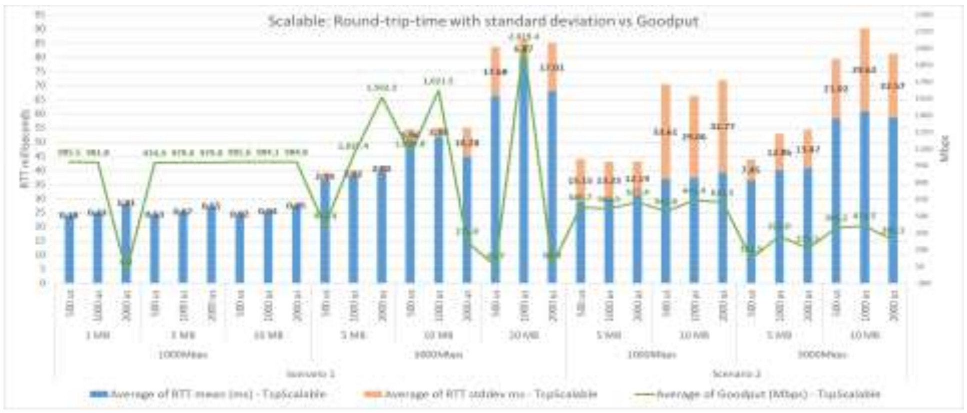


Figure 5.3.11: TCP Scalable's Round trip time (RTT) and goodput.

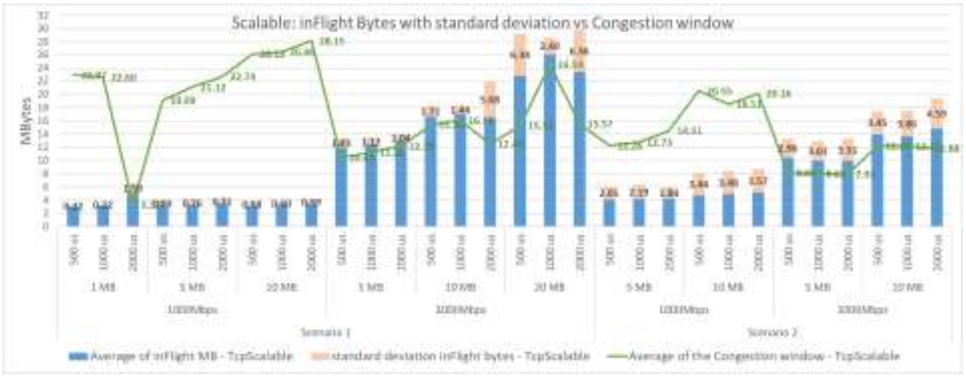
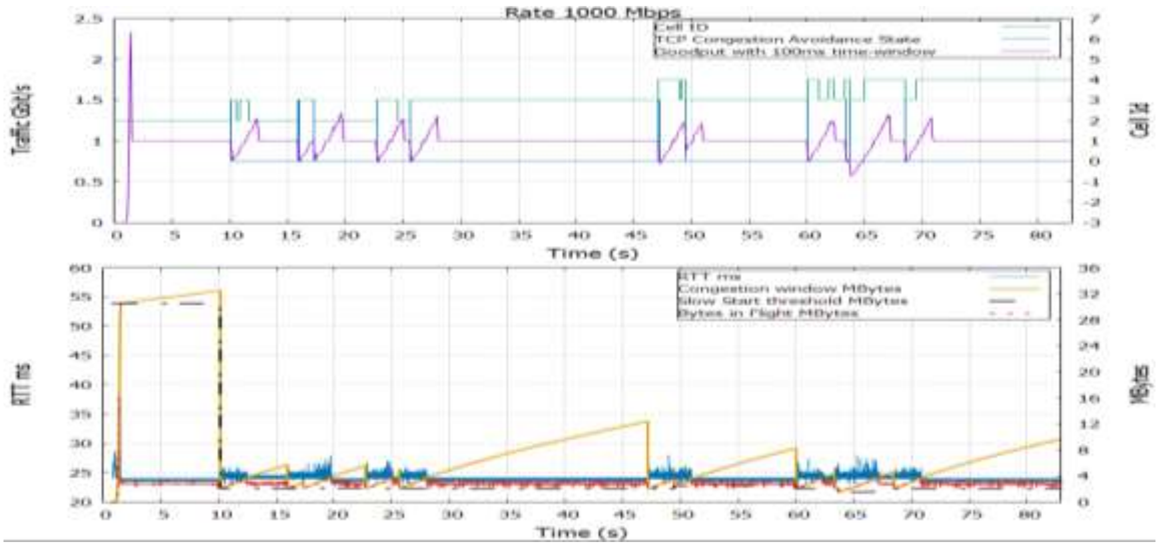


Figure 5.3.12: TCP Scalable's inflight Byes (blue bars) vs Congestion Window (green line)

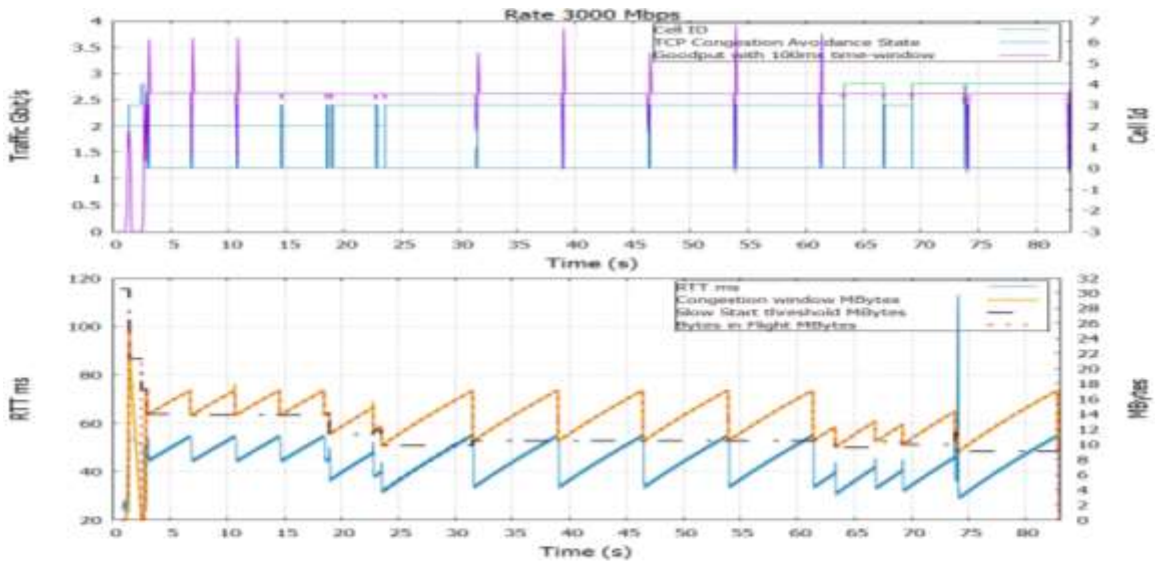
5.3.5. HighSpeed

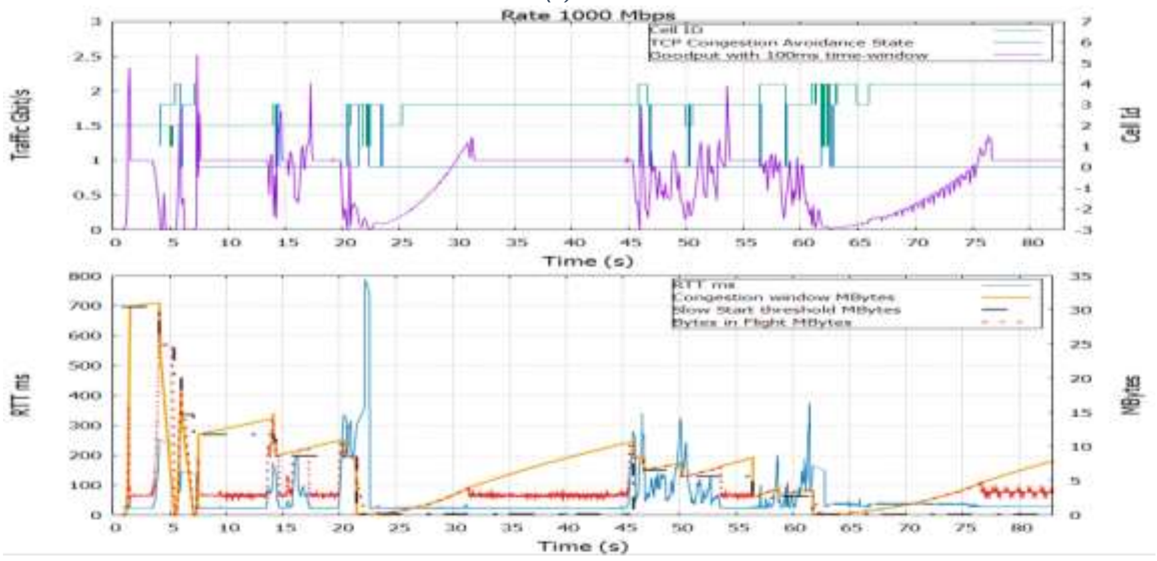
The impact of TCP Highspeed on the performance of Dual Connectivity (DC) system is shown through Figure 5.3.13-Figure 5.3.15.

In Figure 5.3.13, the simulation with Highspeed shows an improvement compared with the previous algorithms. The increment of the $cwnd$ will be done adaptively depending on the number of segments in the current congestion window, by using the $lookuptable\ A$ or $lookuptable\ B$. In this way the system performance is more stable since it uses an adaptive incremental approach, as we can see on Figure 5.3.13 for test A and test B respectively. In scenario 2, we can also see that Highspeed improves the stability compared to Scalable. But the system is still undergoing the delay caused by blockage that HighSpeed cannot foresee which in test D, from second 45, induced the system to stay in the Recovery phase till it finishes to retransmit all the missing packets. In addition, after $cwnd$ reduction the throughput needs some time to return to the target speed in tests C and D (urban scenario simulations).



(a) Test A


(b) Test B


(c) Test C

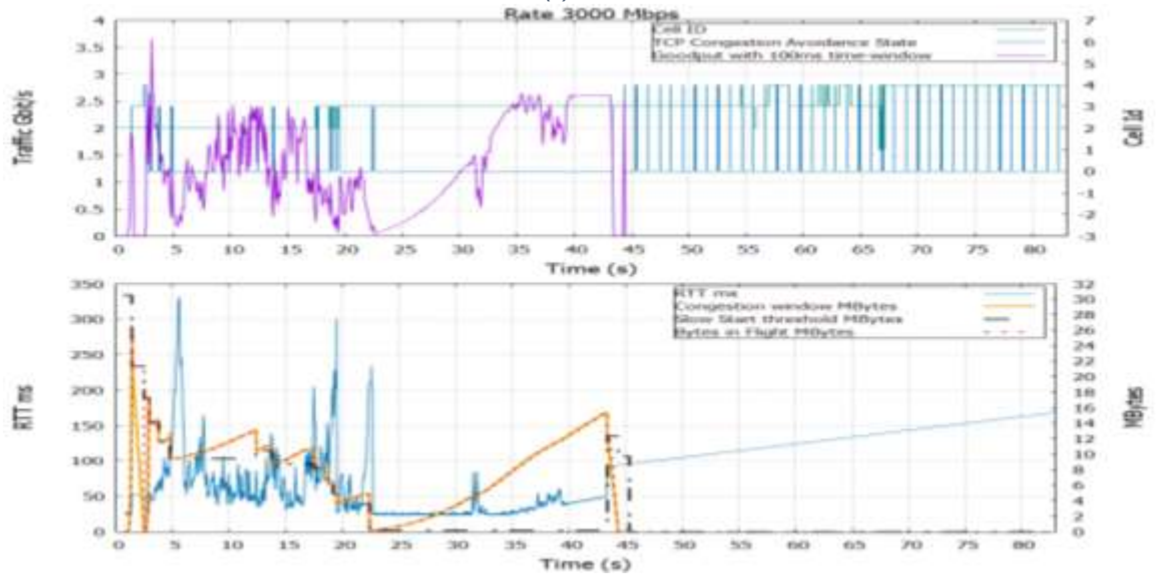

(d) Test D

Figure 5.3.13: TCP HighSpeed performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

The Figure 5.3.14 and Figure 5.3.15 show the result of the simulations done for Highspeed. We can observe that RTT is stable and as we increment the complexity of the system, the standard deviation increments. Due to channel conditions the node degrades the MCS, which reduces the rate. As we can see, the medium behaves as a bottleneck for the system. As a result, the throughput is right away affected. In addition, the mean congestion window is reduced compare to scalable for test A and C at 1000Mbps speed but still leave some unutilized space and for the maximum capacity simulations (Test B and D) the $cwnd$ goes along with the Bytes in-flight to reach the speed target (3000Mbps). As a result, Highspeed set a stepwise increasing function of $cwnd$, and a stepwise decreasing function of $cwnd$, but its convergence speed is very slow in urban scenarios (scenario 2).

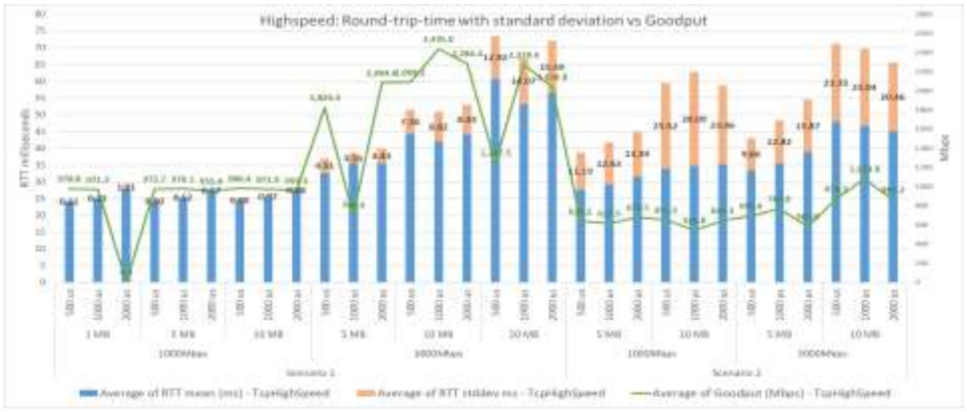


Figure 5.3.14: TCP HighSpeed's Round trip time (RTT) and goodput.

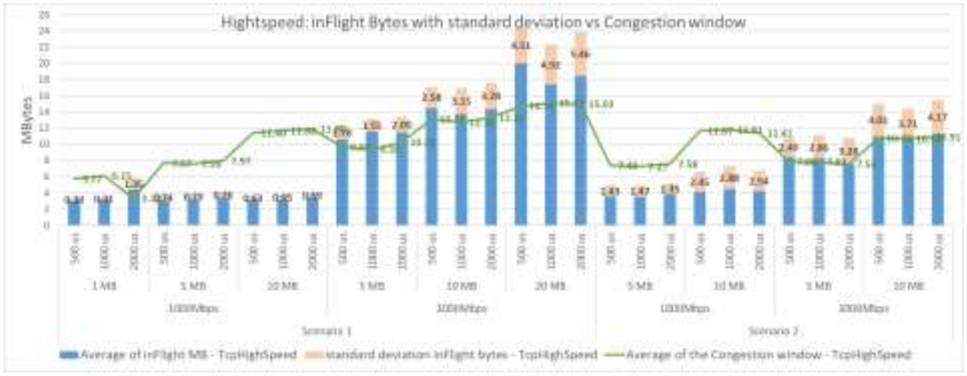


Figure 5.3.15: HighSpeed's inflight Byes (blue bars) vs Congestion Window (green line).

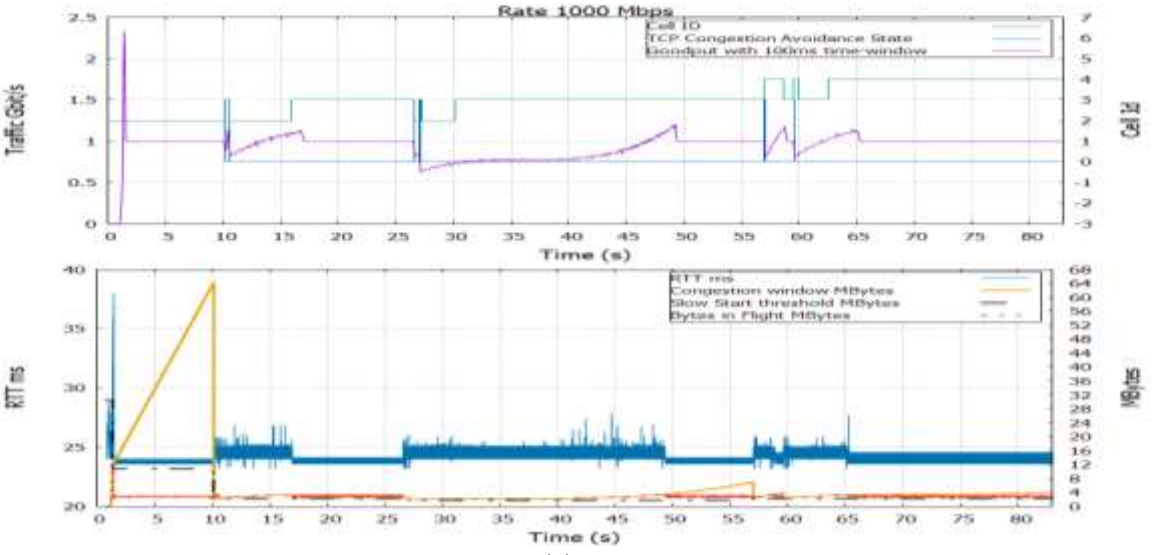
5.3.6. Cubic

The impact of Cubic variant on the performance of mmWave networks is reflected in Figure 5.3.16.

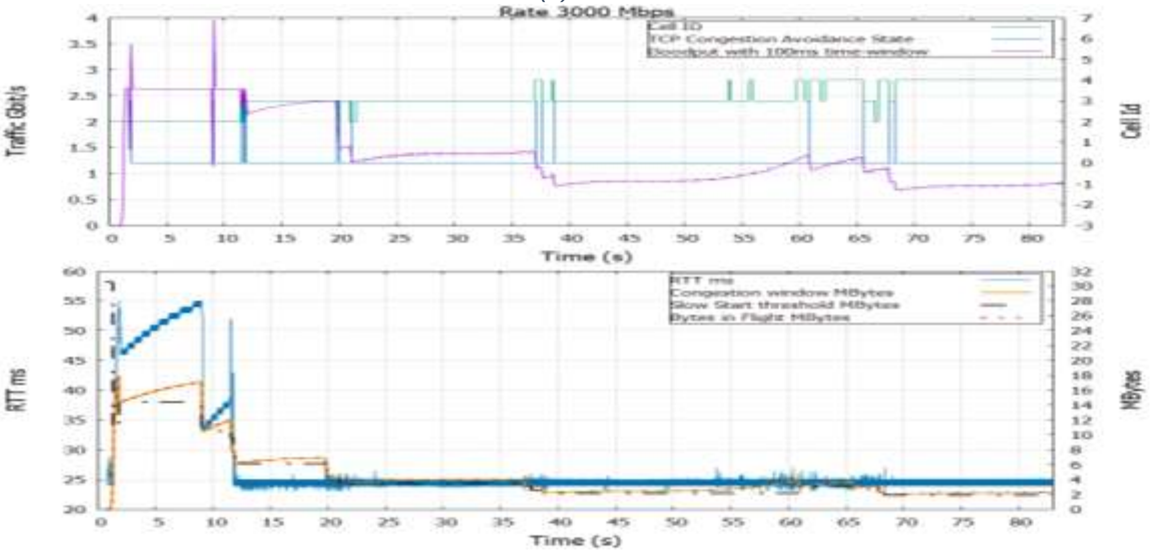
In test A, we can see that after congestion events (CA_DISORDER) $cwnd$ is reduced to the number of Bytes in-flight, which is less aggressive reduction metric and does not alter much the flow of packets in the network. In test B, the system is trying to explore its maximum capacity, so the probability of packet loss or arriving in disorder increments, but Cubic proves to be a stable algorithm. However, after consecutives handovers the $cwnd$ is constantly reduced before it returns to the previous $cwnd$. Because of that the system quickly reaches the previous window size but it spends too much time at the plateau region before it starts to explore higher bandwidths.

In simulation C, the channel is highly degraded in NLOS so in areas in-between nodes the system lacks a dominant signal. The mmWave channel interfere each other and the MCS is reduced, Figure 5.3.16. This increments the bytes occupancy at the RLC layer thus the latency goes to up to 250 milliseconds in the first half of the simulation. In the second half the slow start threshold and the latest $cwnd$ values are already under 5Mbytes, this means that Cubic quickly reaches the previous $cwnd$ however it still needs to stay at the plateau region for around 15 seconds. Another thing observed was that the UE in outage performs a Fast switching to the LTE eNB in the second half of the simulation at 60th seconds, which convey to more delay.

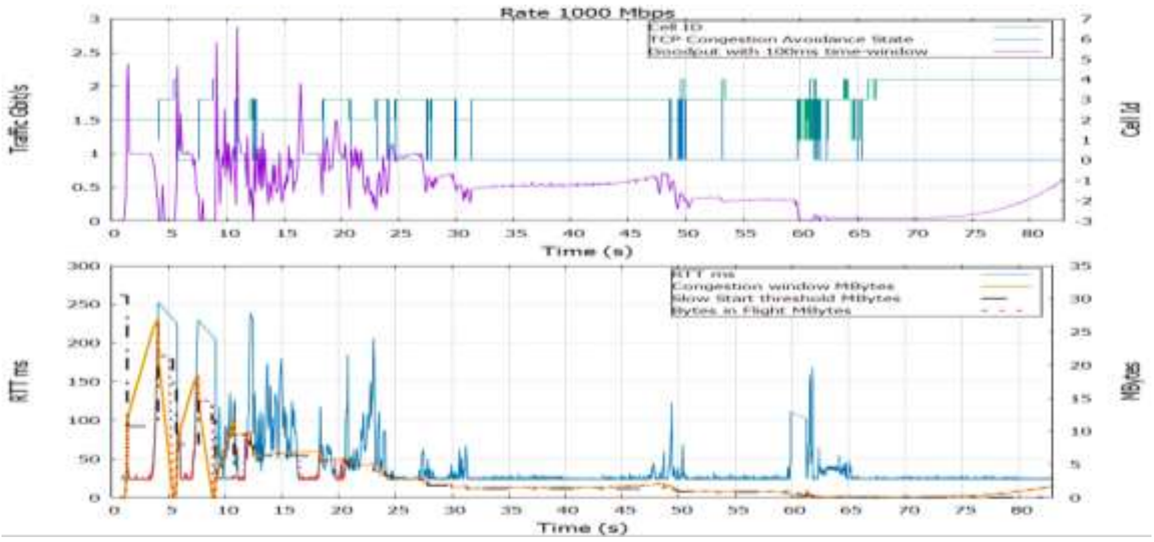
For simulation D, the hybrid slow start algorithm foresees a congestion and reduces the $ssThresh$ and moves to the congestion avoidance phase. However, the NLOS increments the latency because the channel is degraded so the MCS index is reduced thus the Transport block size is smaller. This means that the latency increments and the traffic slows down from second 27 to 45. In this case again the plateau region limits the increment of the rate, since the algorithm isn't aware of the current stability of the link (a clear dominance signal to trying so no delay in the channel).



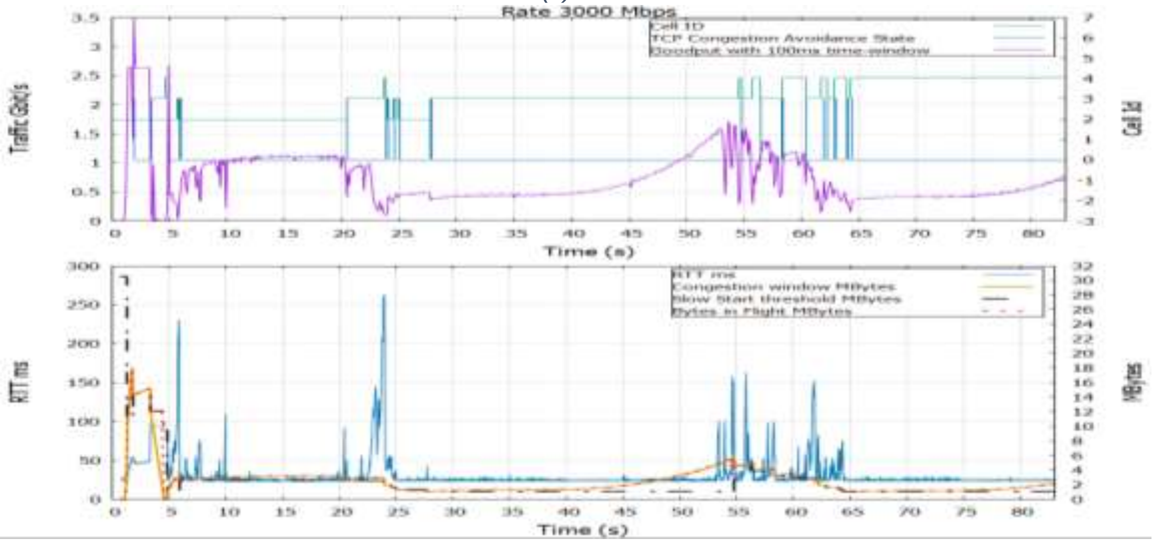
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.16: TCP Cubic performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

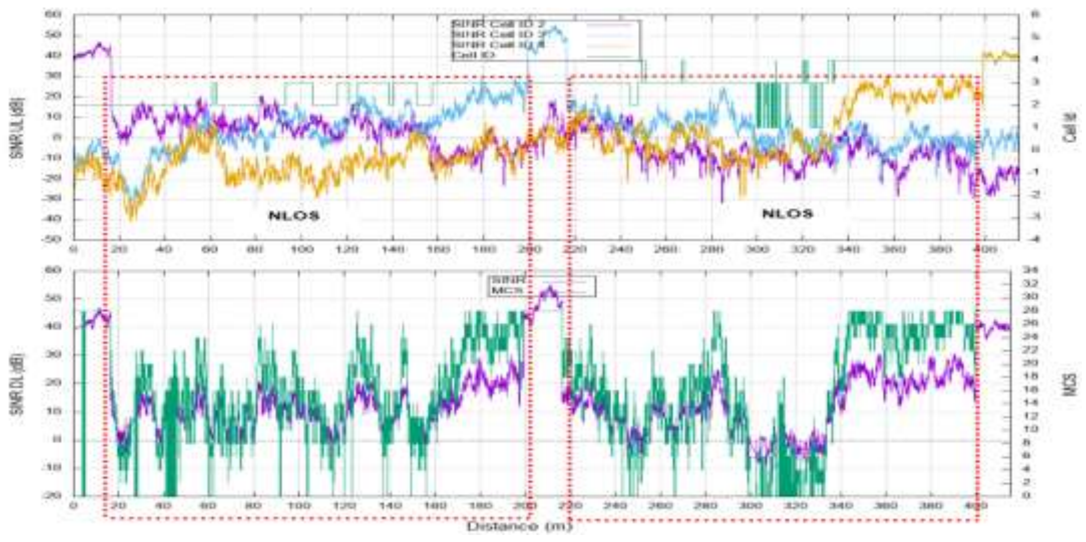


Figure 5.3.17: TCP Cubic SINR and MCS in test C. the red dots enclosed the NLOS area in-between mmWave nodes.

To summarize the results from Cubic, Figure 5.3.16(c) and 5.3.17 show the algorithm performance over the different parameter configurations that were depicted in Table 5.3. We can see that in free-space scenario Cubic is very efficient, in case of small buffers with an X2 latency of 2000 microseconds Cubic was the only algorithm that tackles the congestion because of its $CubicReset()$ method that in case of timeouts it resets the Cubic variables. As observed in Figure 5.3.18 and Figure 5.3.19, the congestion window growth is around 10 times the Bytes in-flight for tests A and C. Because after the system reaches the target of 1000Mbps, the application keeps that speed but the algorithm continuous increasing the $cwnd$. In Tests B and D, as the system goes for the maximum capacity, $cwnd$ and Bytes in-flight grow together and are more affected by congestion events.

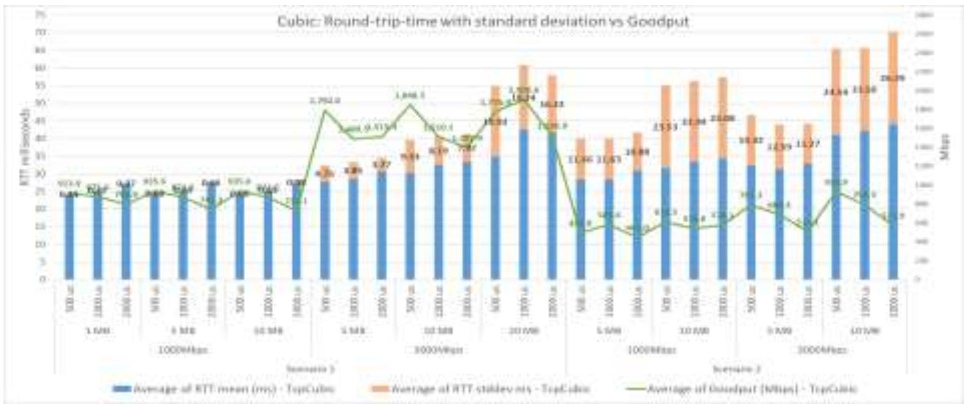


Figure 5.3.18: Cubic's Round trip time (RTT).

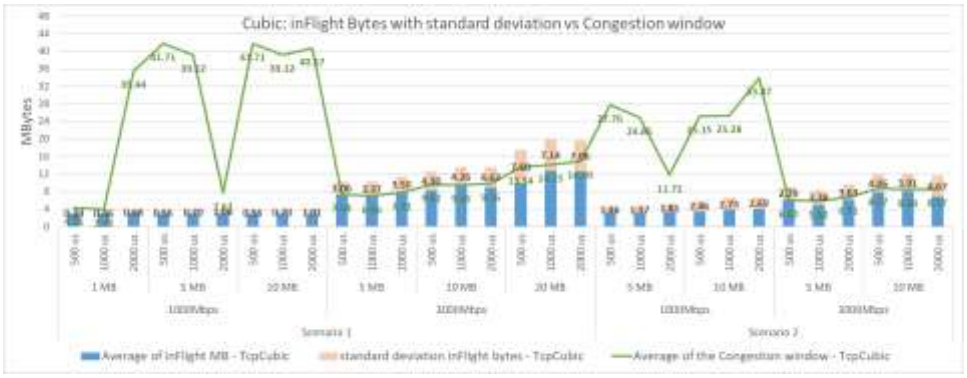


Figure 5.3.19: Cubic's inflight Byes (blue bars) vs Congestion Window (green line)

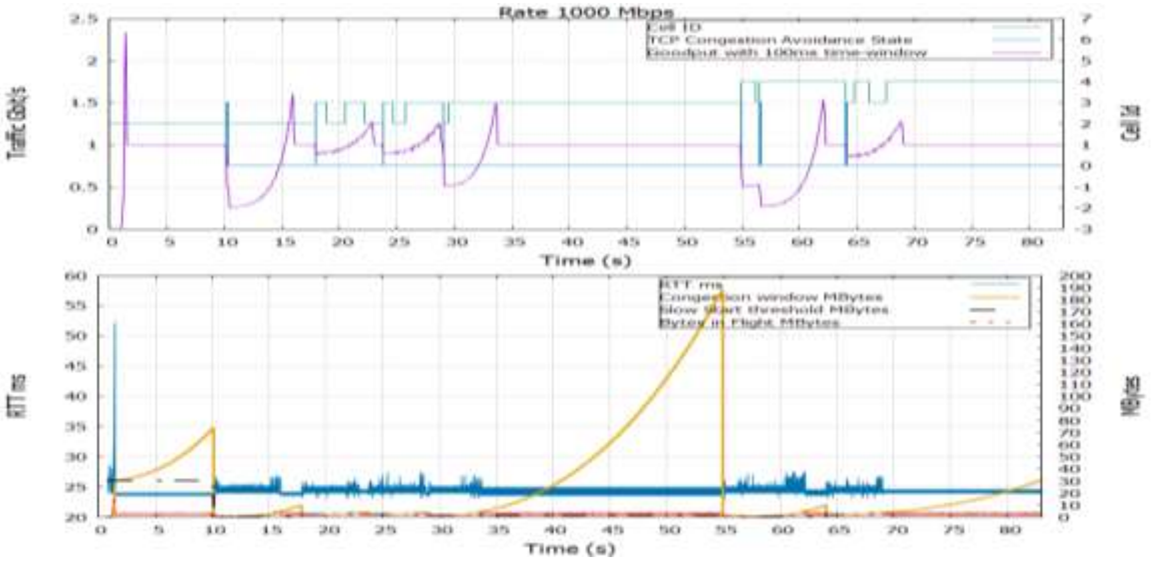
5.3.7. HTCP

The impact of HTCP variant is exemplified in Figure 5.3.20 and the overall result from the simulations stated in table 5.3 are shown in Figure 5.3.21 and Figure 5.3.22.

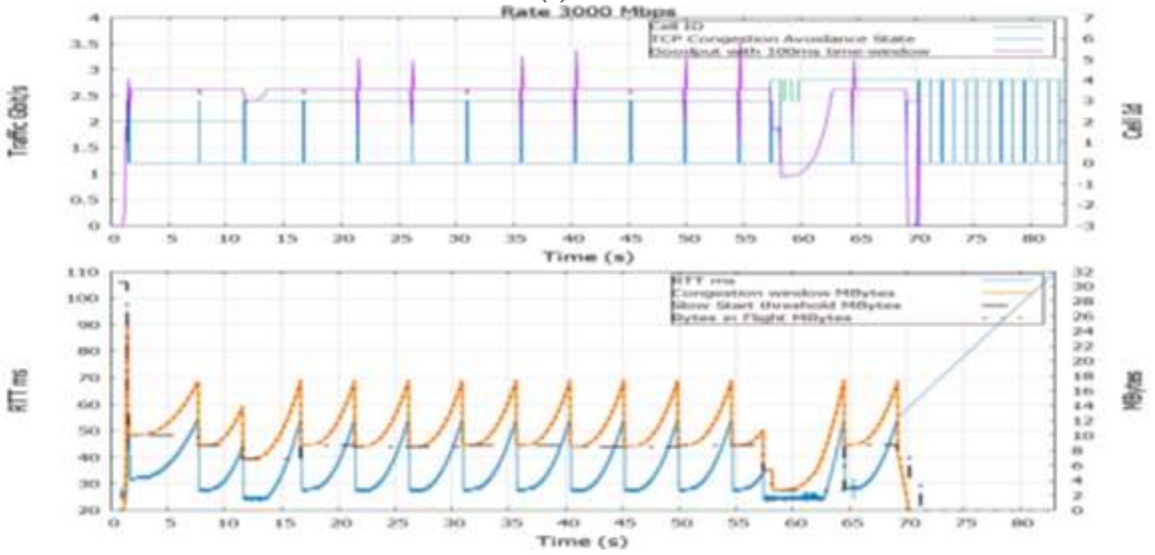
We can see that the $cwnd$ has a curve shape of the quadratic alpha function reaching greater speeds as shown in Figure 5.3.20. In test A, after the congestion event, the $cwnd$ grows dramatically, generating drops without handovers. Though, it returns quite fast to the target speed. In fact, during the recovery phase, the slow start threshold is set to the number of Bytes in-flight (the $cwnd$ is set to the current number of bytes in-flight).

The situation can be better noticed in Test B, where the system is going to the maximum capacity achievable. The increment trend of the algorithm fosters spontaneous congestion events that ends in packets dropped. Another thing which is being observed is: since the congestion window increases, the RTT increments proportionally. This means long latency, as we can see, RTO events start to appear in the last thirteen second of the simulation (test B).

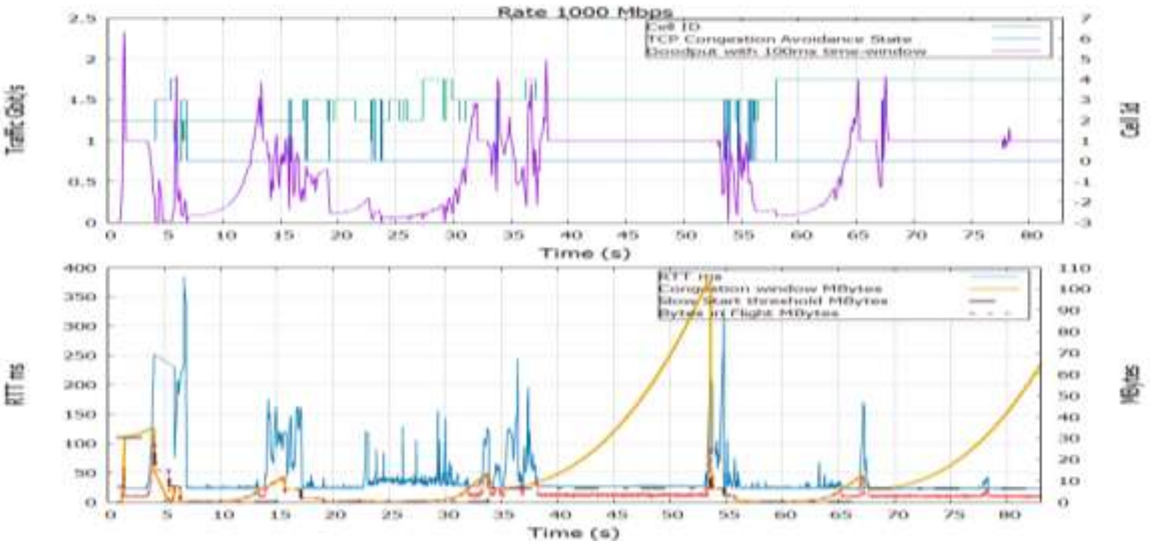
Tests C and D show the aggressive nature of the algorithm which produces constant $ssThresh$ reduction. Because of the building blockage effect, the traffic grows slowly and once it reaches its maximum it drops because of the increment trend of the algorithm. This algorithm increments the probability of packet lost.



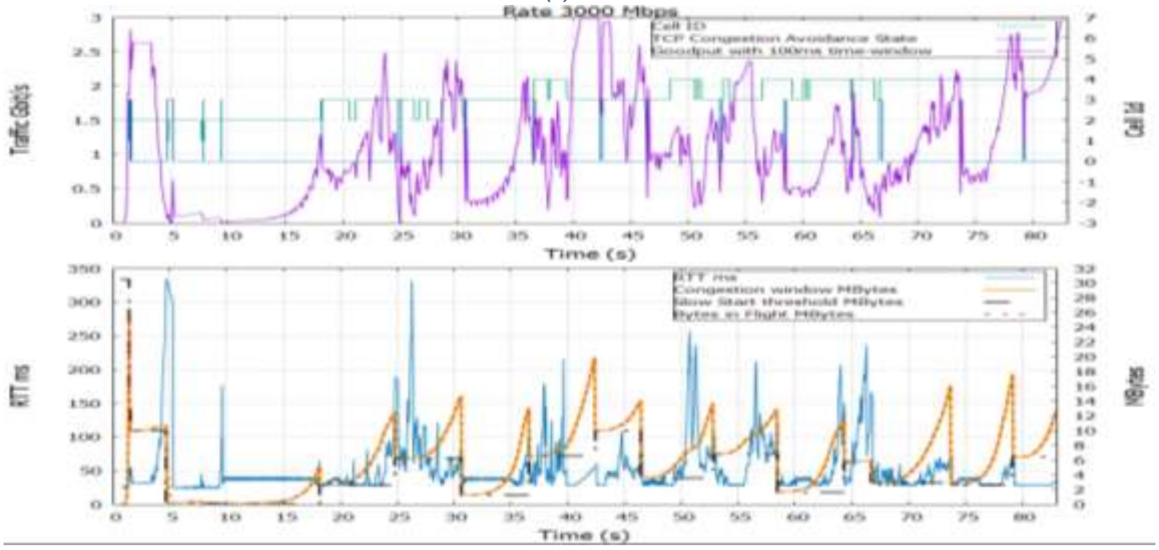
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.20: HTCP performance RLC Buffer =10 MB, X2 Latency =500 microseconds

HTCP aims at a faster convergence and better utilization by setting α to be an increasing function of the time elapsed since last back-off and setting β to be such that the link is always around full utilization, even after the back-off.

The figure below shows the round-trip time versus the goodput, for the different parameter configurations defined. As we can see in test A, the roundtrip time does not vary much compared to what the graphs shows later in test B, where the standard deviation increments and therefore the round-trip time increments as well. This is because of the constant congestion events. Another thing observed: as we increment the X2 latency the goodput is nearly 0. For Figure 5.3.22 the number of Bytes in-flight goes along with the congestion window. It means that the system uses the maximum capacity.

The overall result show that HTCP is very aggressive in congestion avoidance phase and slightly decrease the $ssThresh$ to the number of bytes in-flight time with a β function that depends on delay since last congestion event and throughput variation. The success of these solutions depends on how accurately α and β considers different conditions.

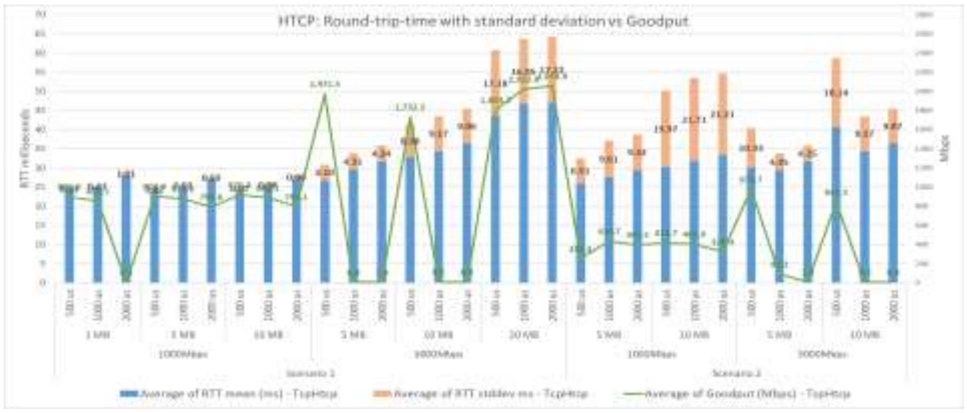


Figure 5.3.21: HTCP's Round trip time (RTT).

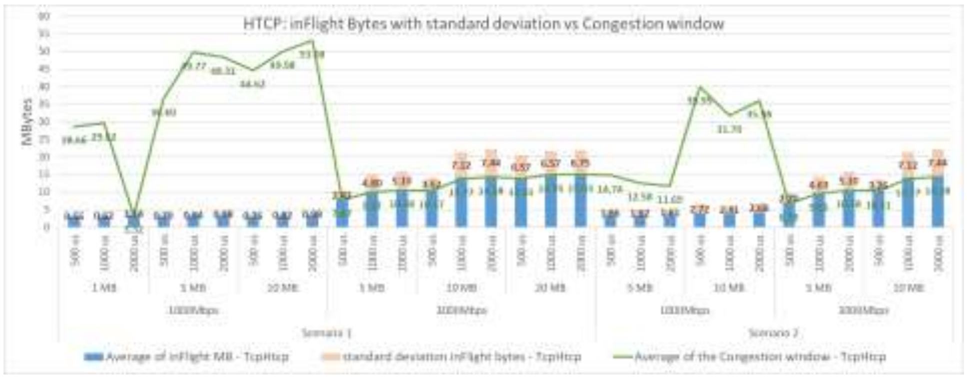


Figure 5.3.22: HTCP's inflight Byes (blue bars) vs Congestion Window (green line)

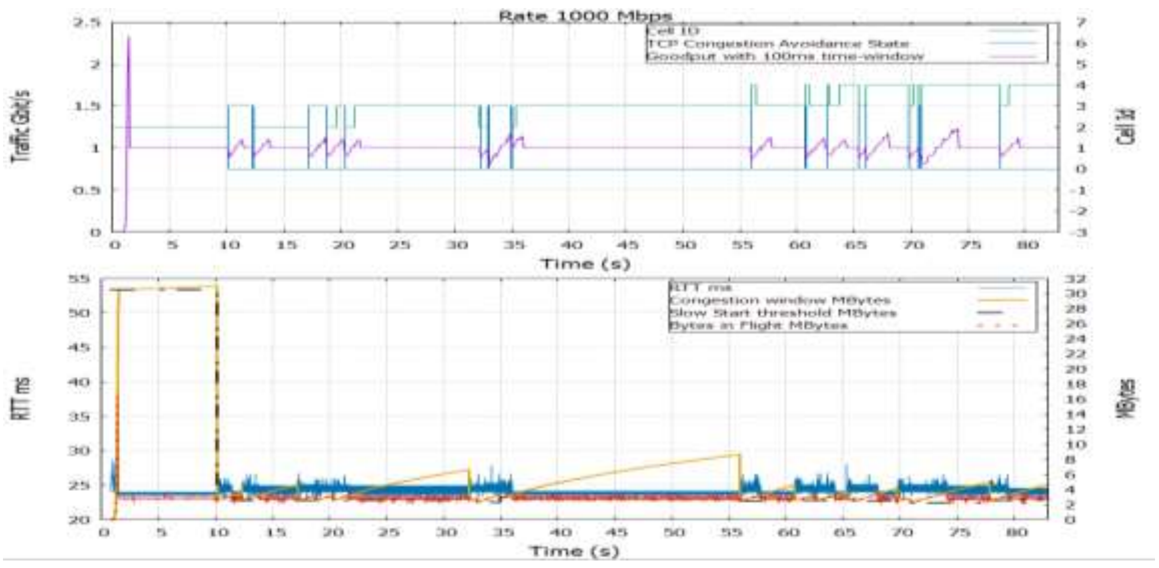
5.3.8. Illinois

The results of the impact of Illinois variant is shown in Figure 5.3.23 for each of the test types. The congestion window shows a concave pattern, it grows fast as far from congestion and slowly as it approaches to congestion (estimation based on RTT jitter).

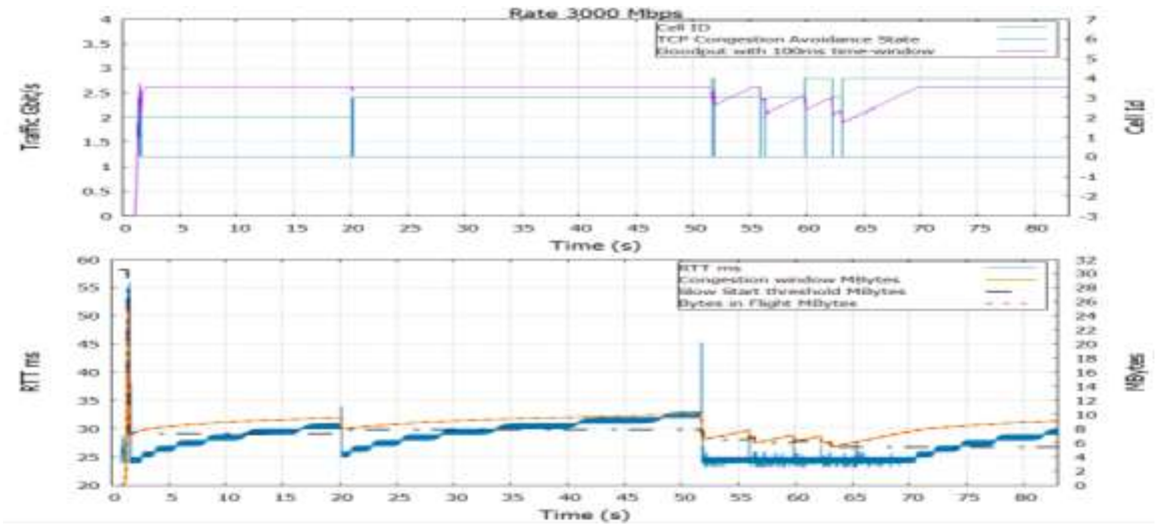
The Illinois simulation shows a good performance. As we can see in test A, after the congestion event the goodput is not altered much.

The congestion window adaptively grows depending on the RTT values. It can be better seen in test B, where the congestion window, after drop event, grows in a concave shape. This means that $cwnd$ slows down when the latency increments. This approach guarantees a maximum channel utilization.

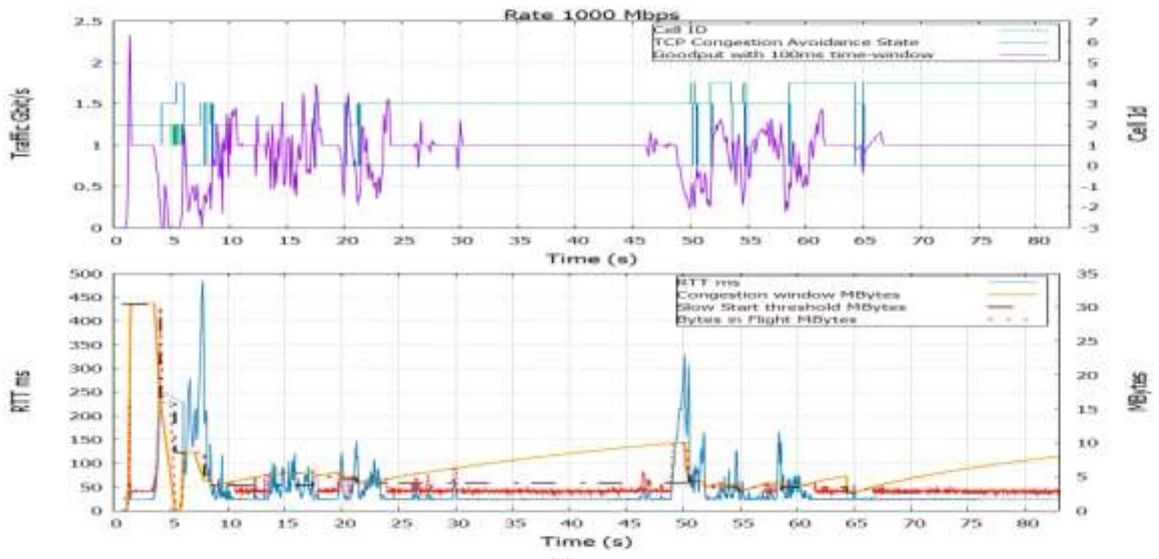
Where we have blockage, the channel is better used since the congestion window starts growing faster after the drop event and slower in the last part.



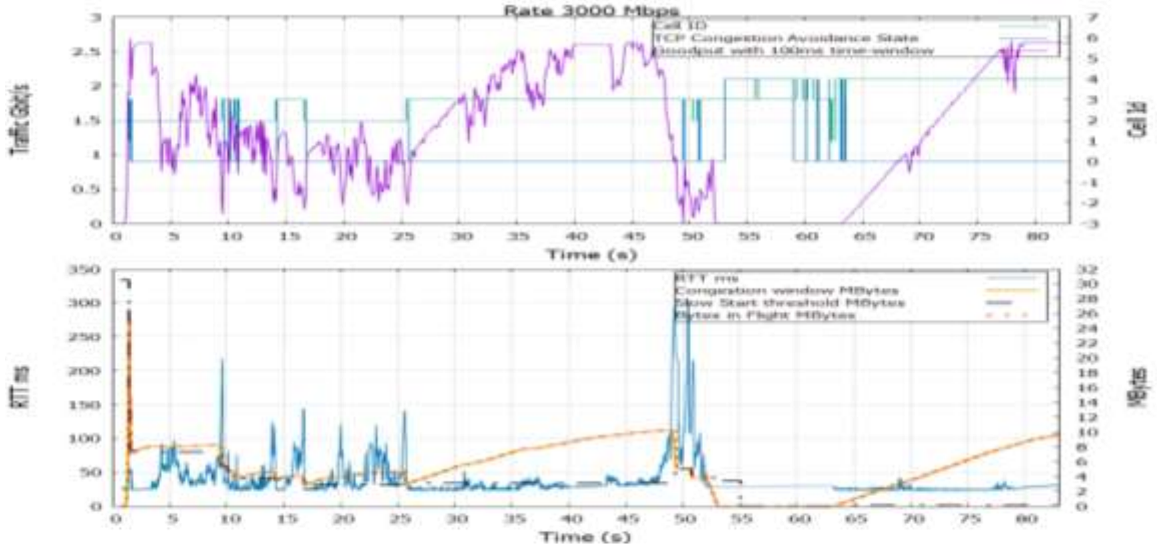
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.3.23: TCP Illinois performance RLC Buffer =10 MB, X2 Latency =500 microseconds.

What calls more the attention is that the RTT standard deviation is reduced, so it means that the system is more stable compared to the previous algorithm mentioned, as we can see in Figure 5.3.25. The number of Bytes in-flight is also reduced, but it still shows some resistance when X2 latency is set at 1000 ms and 2000 ms, where the goodput is around 4 Kbps.

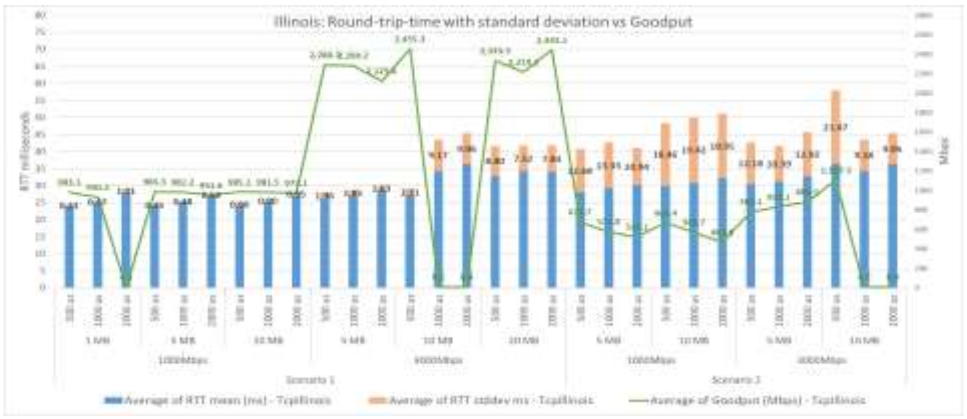


Figure 5.3.24: Illinois's Round trip time (RTT).

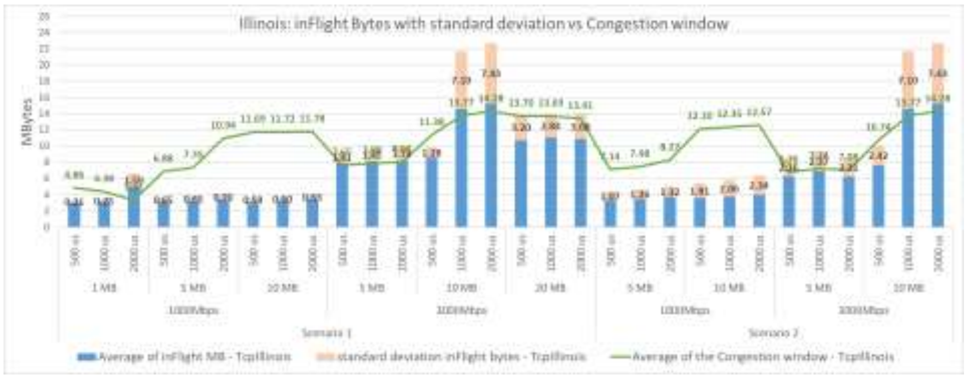


Figure 5.3.25: Illinois's inflight Byes (blue bars) vs Congestion Window (green line)

5.4. Performance Analysis

In this work, the TCP performance metrics are based on the user experience. In this study, we choose throughput (goodput) and Round-Trip Time (RTT) as the main performance metrics to compare the performances of seven TCP congestion control algorithms (CCAs) on LTE/5G mmWave DC system.

5.4.1. Slow start phase effect

The standard slow start mechanism increments the $cwnd$ by one segment per ACK. This allows the system to quickly increment the amount data, it is very helpful especially for BDP network, but it is also aggressive since the sender doubles the number of segments per ACK. This means that in the last part of the slow start phase, the double factor overflows the buffer or generates RTT spikes.

For example, if the sender previously sent 1000 packet (1000 packets x 1400 Bytes=1.4MBytes), later after the reception of the ACK the sender is going to double the number of packets to 2000 so the buffer occupancy will be 2.8 Mbytes and so on. Some simulations were done to encounter this effect, in test A we set up the RLC buffer to 1Mbytes with varying X2 latency. In slow start phase, the CC algorithms result in congestion before they reach the $ssThresh$ value.

The results in section 5.3 showed that for small latencies, at the X2 link, such as 500 and 1000 microseconds the algorithms overcome the high traffic performing fast retransmit. Although, when the X2 link is set at 2000 microseconds the packets in-flight increment so the RLC occupancy increases along with the probability of overflow. The RLC buffer employs a Drop-tail queue mechanism. When a queue becomes full, packets are simply dropped. Since packets can be dropped on TCP connections, the sender can be forced to go into slow-start mode or continue in fast retransmission till the receiver finish requesting the dropped packets.

Among the CC mechanisms the only one that overcomes these difficulties is Cubic. Considering the effect of the slow start phase, the hybrid slow start mechanism implemented in Cubic is the most promising solution thanks to its reset method that in case of congestion the Cubic parameters are reset.

5.4.2. Continuous timeout events

Another observation found was that sometimes in retransmission phase the packets run out of time to be acknowledged. The sender continues in retransmission phase.

In Figure 5.4.2 the RLC buffer is set at 1 Mbyte, the memory overflow due to the traffic flow and queuing management algorithm implemented (Drop-tail) discard the arriving packets. The user sends duplicated ACKs to the sender which moves to retransmission mode. The sender starts forwarding the unacknowledged packets but those arrive in delay, caused by delay in the queue, large X2 latency and initial slow start threshold, generating RTO events and reduction of the congestion window to one packet length. Once the system arrives to that point it turns almost impossible to restart the transmission.

It’s been observed that even though the congestion window was reduced, the receiver keeps sending duplicate acknowledgement of the packets that arrives after. The sender timeouts the packets that where lost in the RLC buffer but because it receives ACKs of the others packets that successfully reach the receiver the sender remains in Retransmission phase till it finishes to transmit the unacknowledged data.

In other words, even though we've timed out and we're starting to retransmit all the packets, it immediately aborts slow start phase and go into fast recovery again because of the state's transition, Figure 5.4.1.

The reason that it holds back further fast retransmissions is that some retransmissions after a timeout will cause duplicate ACKs. Because they are retransmissions of segments that have already been received. If the sender repetitively goes back into fast retransmit due to several gaps in the receiver buffer, $ssthresh$ will be reduced to one segment size. In high bandwidth-delay product networks, the TCP flow is somewhat damaged because the only way to rebuild the window is through congestion avoidance phase. This is due to the large buffers, there are still duplicate ACKs in flight, received after the initial RTO. These are causing the TCP state machine to go right back into fast retransmit after the timeout, even though it probably should continue slow start. And somewhere along the line.

This means that the cost of not being able to detect the real retransmissions is the reduction of the $ssThresh$ to one segment size.

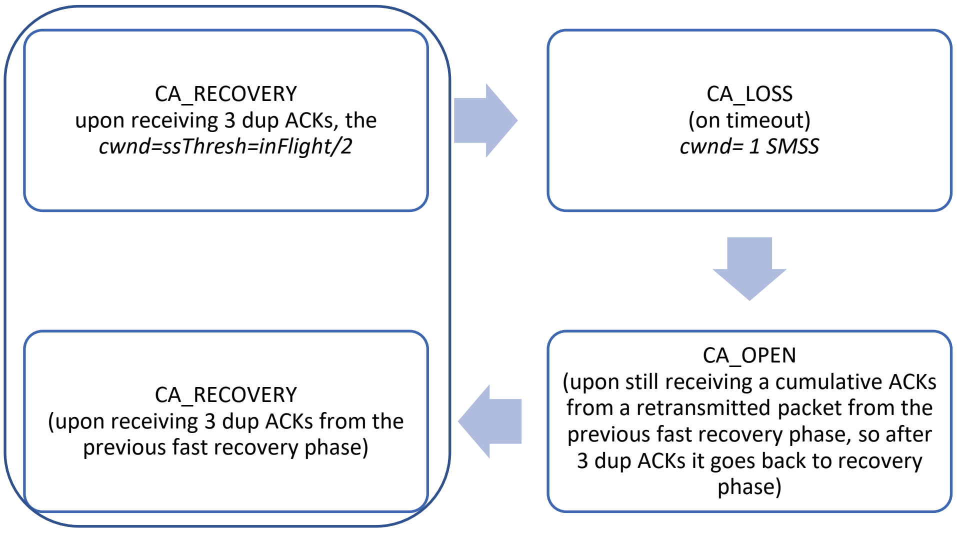

CA_RECOVERY
upon receiving 3 dup ACKs, the
cwnd=ssThresh=inFlight/2
CA_LOSS
(on timeout)
cwnd= 1 SMSS
CA_OPEN
(upon still receiving a cumulative ACKs from a retransmitted packet from the previous fast recovery phase, so after 3 dup ACKs it goes back to recovery phase)
CA_RECOVERY
(upon receiving 3 dup ACKs from the previous fast recovery phase)

Figure 5.4.1: Persistent RTO events, sequence of the states.

Among the TCP congestion algorithms, it's been observed that cubic handle this problem by multiplicative decrease of the $cwnd$ (1-beta, beta =0.8), reducing the $ssthresh$ to the $cwnd$ and when an RTO event occurs the protocol restarts to slow start, with function $cubicReset$ resetting the parameters to 0 but the $ssthresh$ remaining with current value. Since the new $ssthresh$ is smaller the system quickly achieves the $ssthresh$ and move to congestion avoidance.

The figures below show the scenario mentioned above, in Figure 5.4.2 we can see New Reno's performance (all the CC algorithms show this behaviour but Cubic does something different under the same parameter configuration) that upon the constant RTO events the $ssThresh$ shrinks to almost nothing and below the RLC layer that maps the packets transmitted and the retransmitted during RTO events (from second 2 till the end). Figure 5.4.3 shows the performance of Cubic for the same configuration parameters that withstand the small buffer size for 1Gbps goodput.

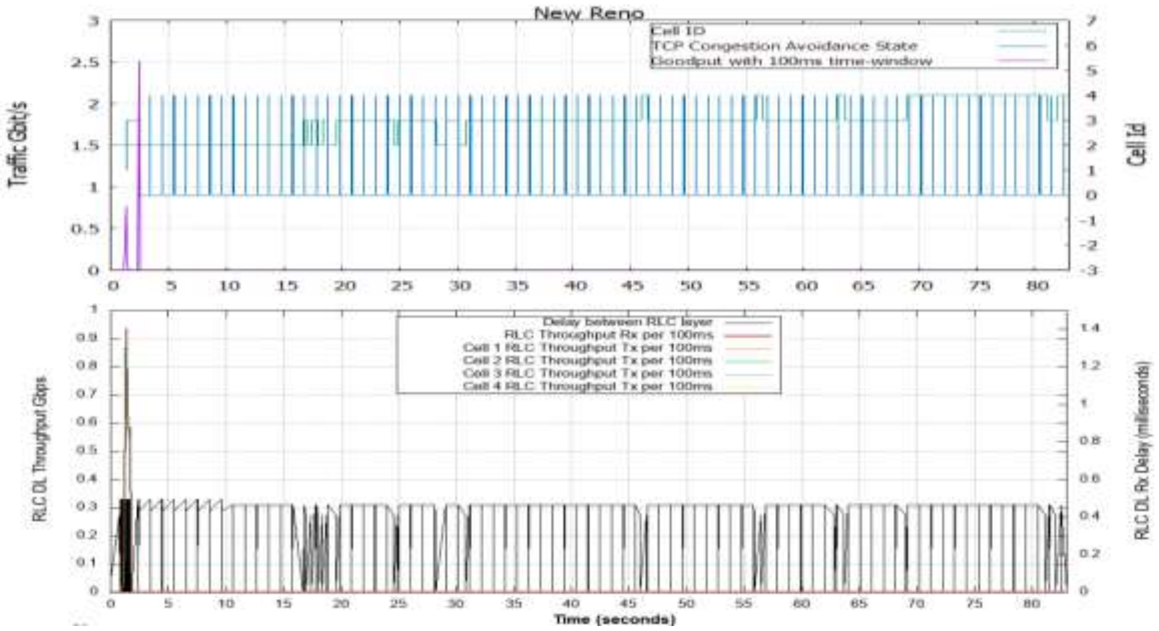


Figure 5.4.2: NewReno's TCP (top) & RLC layer (bottom) performance, RLC buffer=1MB, X2 Latency 2000us and rate = 1Gbps

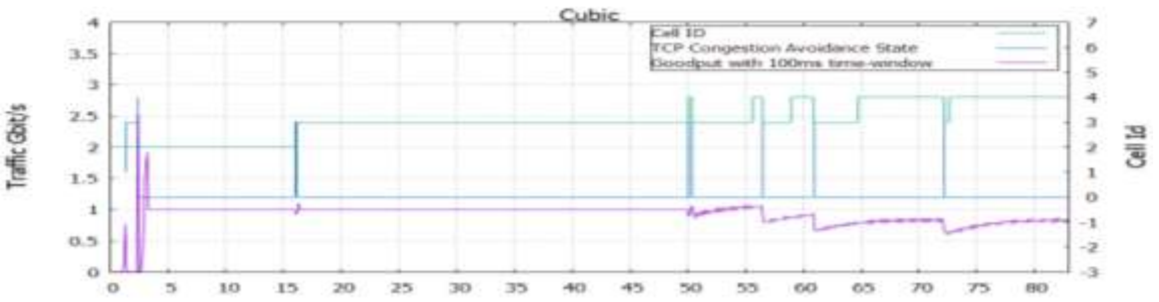


Figure 5.4.3: Cubic's performance, RLC buffer=1MB, X2 Latency 2000us and rate = 1Gbps

5.4.3. Handover effects

During the simulation campaign, it's been observed that most of the congestion events were caused by handovers. In this subsection, we discuss the effects of the handover implemented.

The secondary cell handover and fast switch are lossless handover procedures that start at the LTE eNB coordinator, the data session is forwarded between mmWave nodes through the X2 link. During a dynamic time to Trigger (TTT) interval the coordinator checks the user link before it decides whether the user remain attached to its current cell or initiate a switch. The decision algorithm considers how the different SINR, from Report Tables, behaves during a varying time window (TTT). Indeed, mmWave systems are very susceptible to the channel conditions and blockage. That's why the use of dynamic TTT helps to adjust the timers in order to reduce the ping pong effect observed in section 5.3, where we showed the performance per each CCA.

When the target mmWave gNB of a HO receives the source node buffer and the incoming packets from the TCP application, what the target node receives are the not the transmitted PDCP SDUs and the PDCP retransmission queue. This means that all the data frames are unnecessarily retransmitted reducing the losses but incrementing the TCP throughput. Thus, the impact at the sender is reflected by an increment on the RTT. If we return to section 5.3, we can observe that after congestion events caused by HOs the RTT doesn't drop immediately, instead it reaches values in the order of hundreds of milliseconds because the packets were just forwarded but not dropped.

It might also happen that the target buffer overflows with the incoming data from two sources discarding packets. Therefore, the TCP sender assumes packet loss event and moves to retransmission phase until the dropped packets are acknowledged. If the delay persists, the RTO event forces the sender to return to slow start.

A congestion control algorithm that some how foreseen congestion and slow down the packet transmission would be needed. Considering this fact, we can see that Illinois is the most promising solution.

5.4.4. Goodput

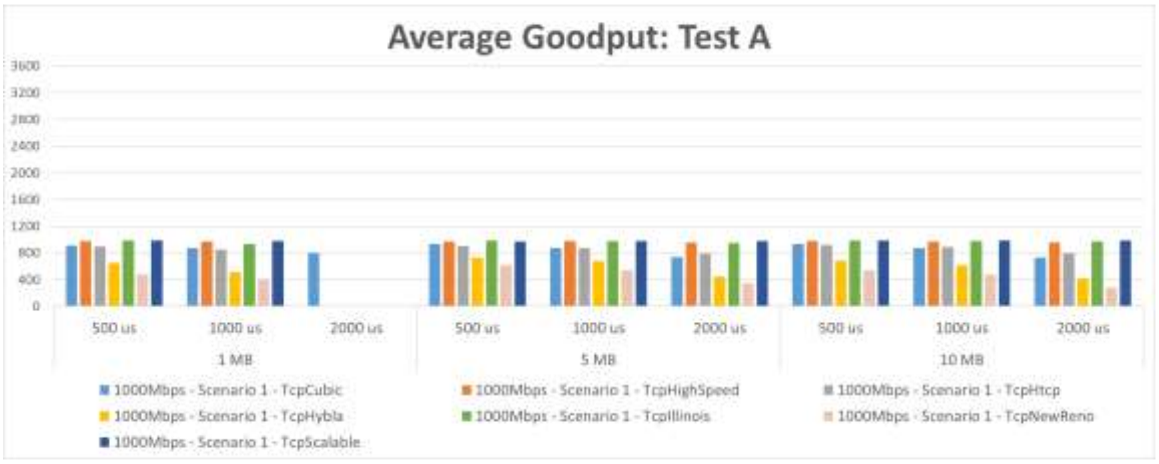
In order to reach high rates, the sender increments the congestion window per round-trip time (RTT) during congestion avoidance. This increment can be multiplicative or additive depending on the implemented congestion algorithm which is going to increment the occupancy of memory on the network elements.

A proper buffer size is important to increase the application layer's data rate. In case the RLC buffer is small, it might overflow and the packet drops, generating RTO or fast Retransmission events. Another scenario is when increasing the RLC size increments the packets occupancy, causing larger latencies in the transmission.

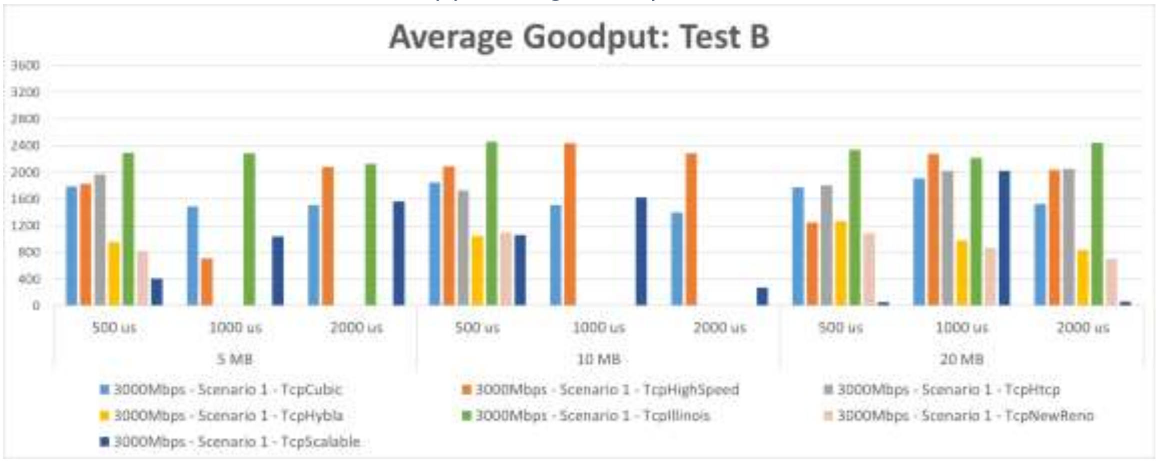
While increasing the X2 latency for different RLC buffer size, we can notice that the overall throughput decreases in most of the cases. These throughput results depend on how each congestion algorithm handle congestion events and their strategy to increase number of packets sent per RTT during congestion avoidance. Another element is the number of Bytes in-flight generated, which are proportional to the rate and to the RTT.

In Figure 5.4.4, it is shown the TCP algorithm performance in 1, 5 and 10 MB RLC buffer for 3 different X2 latencies (500 us, 1000us and 2000us). Cubic, Highspeed, HTCP, Illinois and Scalable algorithms are above the rate target 1Gbps, but New Reno and Hybla are around half of the target. This is caused by the linear increment of their congestion window during congestion avoidance. In case of Hybla, the increment of the congestion window is done by one segment per RTT because this protocol was designed to guarantee transmission for large latencies. Similarly, New Reno's increment is mostly one byte per each acknowledgment received making it almost impossible to return to the target rate.

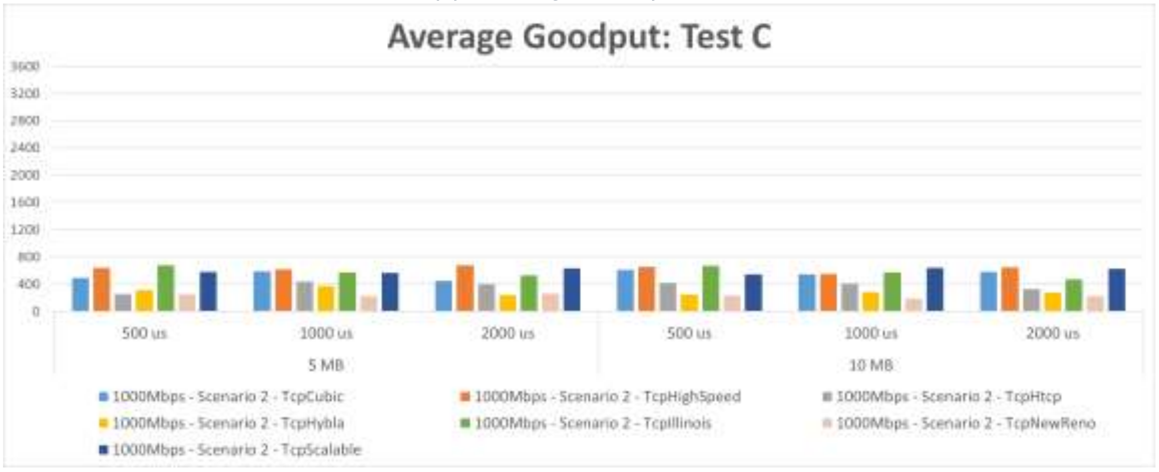
On the other hand, Cubic shows a good result because its algorithm takes into account the time since the last congestion event in window function and set the prior congestion windows as inflexion point for the Cubic function. Meanwhile, the others suffer RTO events caused by increment of Bytes in-flight which overflow the RLC buffer. So, the RLC buffer receives more packets than it can handle and starts dropping. This situation can be solved by expanding the buffer as we can see in Figure 5.4.4. In scenario 2, the level of interference caused by the buildings, that bring LOS/NLOS conditions, reduces considerably the average goodput experience. Because the periods of outage and low modulation contributes to large delay producing packets arriving in disorder or timeout events.



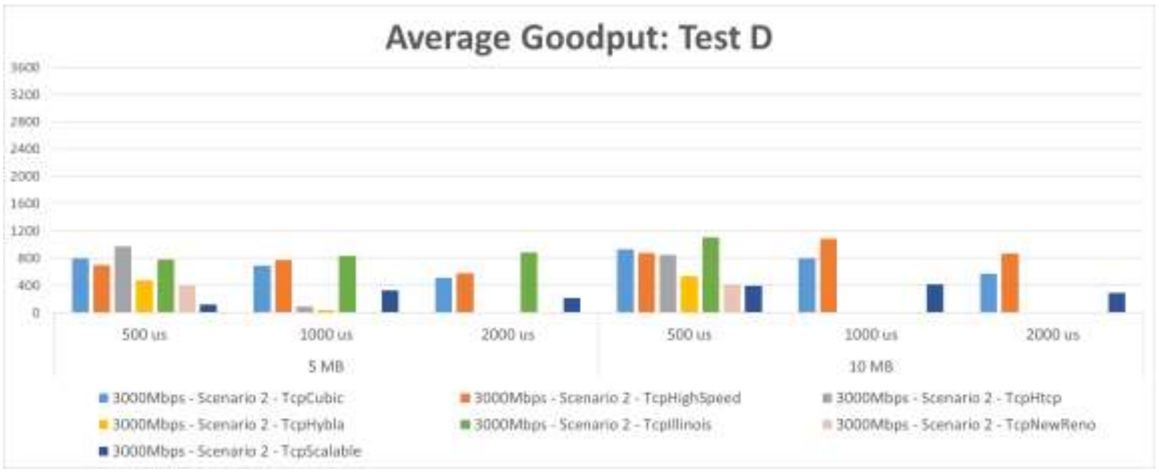
(a) Average Goodput Test A



(b) Average Goodput Test B



(c) Average Goodput Test C



(d) *Average Goodput Test D*

Figure 5.4.4: Comparison of the user goodput experienced for B_{RLC}={1,5,10 and 20 Mbytes} buffer size in varying latencies for every congestion control algorithm.

Table 5.5 shows the average goodput of the simulations performed per set of parameters with the corresponding standard deviation.

protocol	Buffer size RLC	X2 Latency (us)	Rate Gbps	Scenario 1		Scenario 2	
				RTT mean with stddev (ms)	Goodput with stddev (Mbps)	RTT mean with stddev (ms)	Goodput with stddev (Mbps)
Cubic	5 MB	500	1 Gbps	24.09 +/- 0.68	935.62 +/- 16.42	28.35 +/- 11.66	492.86 +/- 37.59
			3 Gbps	27.69 +/- 4.76	1791.97 +/- 60.58	32.38 +/- 14.32	791.34 +/- 72.96
		1000	1 Gbps	25.11 +/- 0.65	871.59 +/- 21.21	28.39 +/- 11.63	582.36 +/- 33.09
			3 Gbps	28.55 +/- 4.85	1489.49 +/- 66.7	31.37 +/- 12.59	686.27 +/- 59.6
		2000	1 Gbps	27.13 +/- 0.66	741.34 +/- 24.9	30.84 +/- 10.88	444.99 +/- 37.64
			3 Gbps	30.68 +/- 3.77	1515.28 +/- 81.91	32.88 +/- 11.27	511.08 +/- 68.01
	10 MB	500	1 Gbps	24.09 +/- 0.68	935.62 +/- 16.42	31.52 +/- 23.53	610.47 +/- 37.22
			3 Gbps	30.23 +/- 9.33	1848.52 +/- 61.71	40.94 +/- 24.54	929.89 +/- 70.13
		1000	1 Gbps	25.11 +/- 0.65	871.59 +/- 21.21	33.34 +/- 22.94	538.83 +/- 39.2
			3 Gbps	32.48 +/- 8.19	1510.09 +/- 86.94	42.12 +/- 23.5	794.3 +/- 70.34
		2000	1 Gbps	27.16 +/- 0.98	730.15 +/- 25.32	34.29 +/- 23.08	578.06 +/- 36.55
			3 Gbps	33.31 +/- 7.97	1397.91 +/- 84.13	43.94 +/- 26.39	573.91 +/- 57.92
HighSpeed	5 MB	500	1 Gbps	24.08 +/- 0.5	972.72 +/- 19.56	27.58 +/- 11.19	639.25 +/- 37.22
			3 Gbps	32.56 +/- 4.51	1825.39 +/- 86.13	33.45 +/- 9.6	697.57 +/- 79.05
		1000	1 Gbps	25.06 +/- 0.52	976.09 +/- 18.05	29.13 +/- 12.63	617.49 +/- 34.48
			3 Gbps	35.5 +/- 3.16	706.01 +/- 75.16	35.52 +/- 12.82	768.79 +/- 81.55
		2000	1 Gbps	27.1 +/- 0.57	954.97 +/- 17.93	31.61 +/- 13.39	678.51 +/- 32.22
			3 Gbps	35.51 +/- 4.43	2084.57 +/- 65.73	38.78 +/- 15.87	580.82 +/- 69.76
	10 MB	500	1 Gbps	24.08 +/- 0.9	980.4 +/- 16.23	33.86 +/- 25.52	655.5 +/- 34.11
			3 Gbps	44.22 +/- 7.3	2089.23 +/- 62.05	47.89 +/- 23.35	874.33 +/- 76.55
		1000	1 Gbps	25.09 +/- 0.97	971.92 +/- 17	34.64 +/- 28.09	545.84 +/- 35.73
			3 Gbps	42.03 +/- 8.82	2435.01 +/- 51.58	46.67 +/- 23.04	1079.76 +/- 82.9
		2000	1 Gbps	27.12 +/- 0.88	959.28 +/- 17.87	35.2 +/- 23.46	645.33 +/- 38.74
			3 Gbps	44.09 +/- 8.93	2286.43 +/- 56.23	45.07 +/- 20.46	866.18 +/- 76.59
Htcp	5 MB	500	1 Gbps	24.08 +/- 0.62	904.93 +/- 23.89	25.98 +/- 6.51	251.3 +/- 39
			3 Gbps	26.76 +/- 4.1	1971.52 +/- 76.95	30.11 +/- 10.33	973.66 +/- 84.8
		1000	1 Gbps	25.12 +/- 0.59	877.58 +/- 27.03	27.52 +/- 9.61	434.73 +/- 41.26

			3 Gbps	29.51 +/- 4.31	6.78 +/- 10.99	29.32 +/- 4.45	91.06 +/- 18.17
		2000	1 Gbps	27.11 +/- 0.59	794.63 +/- 33.63	29.4 +/- 9.32	397.53 +/- 39.71
			3 Gbps	31.67 +/- 4.24	6.98 +/- 11.91	31.67 +/- 4.25	6.98 +/- 11.91
	10 MB	500	1 Gbps	24.08 +/- 0.97	920.25 +/- 22.83	30.21 +/- 19.97	418.71 +/- 39.59
			3 Gbps	32.82 +/- 8.38	1732.16 +/- 96.77	40.57 +/- 18.14	842.25 +/- 85.52
		1000	1 Gbps	25.11 +/- 0.98	888.14 +/- 25.14	31.78 +/- 21.71	406.87 +/- 40.55
			3 Gbps	34.32 +/- 9.17	8.23 +/- 13.09	34.32 +/- 9.17	8.23 +/- 13.08
		2000	1 Gbps	27.11 +/- 0.96	795.09 +/- 33.74	33.39 +/- 21.21	325.43 +/- 40.16
			3 Gbps	36.37 +/- 9.06	8.42 +/- 12.71	36.37 +/- 9.07	8.42 +/- 12.71
Hybla	5 MB	500	1 Gbps	24.11 +/- 0.56	728.4 +/- 24.89	26.04 +/- 8.62	309.98 +/- 30.59
			3 Gbps	24.18 +/- 0.59	954.28 +/- 47.26	28.25 +/- 11.83	476.87 +/- 45.87
		1000	1 Gbps	25.2 +/- 0.6	679.14 +/- 25.05	27.06 +/- 7.16	367.65 +/- 29.17
			3 Gbps	29.51 +/- 4.31	6.78 +/- 10.99	29.1 +/- 4.08	34.46 +/- 11.93
		2000	1 Gbps	27.18 +/- 0.7	444.44 +/- 29.32	29.95 +/- 8.67	239.92 +/- 31.65
			3 Gbps	31.67 +/- 4.24	6.98 +/- 11.91	31.67 +/- 4.25	6.98 +/- 11.91
	10 MB	500	1 Gbps	24.16 +/- 1.1	682.6 +/- 23.6	28.39 +/- 15.36	243.77 +/- 34.2
			3 Gbps	24.26 +/- 1.21	1050.16 +/- 65.91	30.34 +/- 14.52	533.77 +/- 54.59
		1000	1 Gbps	25.18 +/- 1.15	617.27 +/- 25	29.57 +/- 15.38	280.1 +/- 30.78
			3 Gbps	34.32 +/- 9.17	8.23 +/- 13.09	34.32 +/- 9.17	8.23 +/- 13.08
		2000	1 Gbps	27.24 +/- 1.32	416.44 +/- 28.54	32.73 +/- 19.03	275.21 +/- 39.89
			3 Gbps	36.37 +/- 9.06	8.42 +/- 12.71	36.37 +/- 9.07	8.42 +/- 12.71
Illinois	5 MB	500	1 Gbps	24.06 +/- 0.45	985.34 +/- 14.43	28.03 +/- 12.68	673.72 +/- 35.12
			3 Gbps	26.01 +/- 1.96	2288.29 +/- 52.11	30.48 +/- 12.18	780.07 +/- 71.08
		1000	1 Gbps	25.06 +/- 0.48	981.24 +/- 14.41	29.32 +/- 13.33	572.77 +/- 33.19
			3 Gbps	26.93 +/- 1.85	2284.19 +/- 52.33	31.23 +/- 10.39	833.11 +/- 65.26
		2000	1 Gbps	27.06 +/- 0.5	951.55 +/- 17.6	30.12 +/- 10.94	524.05 +/- 32.82
			3 Gbps	28.43 +/- 1.63	2125.59 +/- 57.13	32.74 +/- 12.92	880.49 +/- 71.69
	10 MB	500	1 Gbps	24.09 +/- 0.9	985.06 +/- 14.43	29.95 +/- 18.46	668.38 +/- 32.67
			3 Gbps	26.87 +/- 2.21	2455.34 +/- 39.51	36.47 +/- 21.47	1107.27 +/- 80.21
		1000	1 Gbps	25.11 +/- 0.9	981.48 +/- 14.96	30.76 +/- 19.42	568.7 +/- 34.93
			3 Gbps	34.32 +/- 9.17	8.23 +/- 13.09	34.32 +/- 9.18	8.23 +/- 13.1
		2000	1 Gbps	27.09 +/- 0.95	973.14 +/- 15.59	32.18 +/- 18.95	467.59 +/- 38.89
			3 Gbps	36.37 +/- 9.06	8.42 +/- 12.71	36.37 +/- 9.06	8.42 +/- 12.71
NewReno	5 MB	500	1 Gbps	24.13 +/- 0.6	617.51 +/- 24.64	26.8 +/- 8.85	250.95 +/- 32.93
			3 Gbps	24.19 +/- 0.58	820.16 +/- 56.47	27.58 +/- 11.04	397.5 +/- 51.51
		1000	1 Gbps	25.22 +/- 0.66	546.69 +/- 26.42	27.42 +/- 9.14	225.28 +/- 30.91
			3 Gbps	29.51 +/- 4.31	6.78 +/- 10.99	29.53 +/- 4.31	6.83 +/- 10.99
		2000	1 Gbps	27.21 +/- 0.8	349.86 +/- 30.78	29.4 +/- 8.27	255.57 +/- 31.17
			3 Gbps	31.67 +/- 4.24	6.98 +/- 11.91	31.67 +/- 4.25	6.98 +/- 11.91
	10 MB	500	1 Gbps	24.18 +/- 1.25	537.68 +/- 25.02	28.35 +/- 15.02	229.49 +/- 33.16
			3 Gbps	24.51 +/- 1.39	1096.95 +/- 72.66	31.93 +/- 17.46	411.64 +/- 64.02
		1000	1 Gbps	25.2 +/- 1.31	476.21 +/- 26.31	33.41 +/- 21.68	186.42 +/- 34.64
			3 Gbps	34.32 +/- 9.17	8.23 +/- 13.09	34.32 +/- 9.17	8.23 +/- 13.08
		2000	1 Gbps	27.3 +/- 1.57	284.69 +/- 31.07	32.35 +/- 18.96	221.93 +/- 38.31
			3 Gbps	36.37 +/- 9.06	8.42 +/- 12.71	36.37 +/- 9.07	8.42 +/- 12.71
Scalable	5 MB	500	1 Gbps	24.09 +/- 0.53	974.89 +/- 19.3	28.84 +/- 15.15	580.66 +/- 41.92
			3 Gbps	36.57 +/- 2.39	412.41 +/- 95.09	36.35 +/- 7.45	121.53 +/- 46.06
		1000	1 Gbps	25.08 +/- 0.67	979.01 +/- 16.86	29.76 +/- 13.23	563.53 +/- 44.89
			3 Gbps	37.68 +/- 2.02	1037.44 +/- 103.48	39.99 +/- 12.86	323.05 +/- 58.49
		2000	1 Gbps	27.09 +/- 0.55	979.05 +/- 16.97	30.9 +/- 12.19	627.43 +/- 45.71
			3 Gbps	39.61 +/- 2	1562.16 +/- 118.21	40.73 +/- 13.47	212.53 +/- 55.71
	10 MB	500	1 Gbps	24.08 +/- 0.92	985.59 +/- 14.02	36.81 +/- 33.61	542.62 +/- 41.8
			3 Gbps	50.46 +/- 3.96	1058.79 +/- 106.55	58.34 +/- 21.02	396.23 +/- 68.17
		1000	1 Gbps	25.08 +/- 0.94	984.1 +/- 14.69	37.28 +/- 29.06	641.41 +/- 40.54

			3 Gbps	51.71 +/- 3.39	1621.51 +/- 96.11	60.73 +/- 29.62	414.48 +/- 73.84
		2000	1 Gbps	27.08 +/- 0.85	984.82 +/- 14.04	39.15 +/- 32.77	623.5 +/- 43.07
			3 Gbps	44.63 +/- 10.28	271.87 +/- 31.76	58.53 +/- 22.57	285.12 +/- 59.98

Table 5.5: Average and standard deviation (stddev) of the round-trip time vs Goodput of 10 simulations per set of parameters. The goodput variance is calculated each 100 milliseconds.

5.4.5. Packet Delivery Ratio (PDR)

It's a very common performance indicator in communication networks that represents a degree of success in the packet transmission trough the network. It is the ratio of number of packets received at the destination to the number of packets generated at the source. The performance is better when packet delivery ratio is high.

Since our objective it to analyse the dual connectivity response in TCP, for the simulations the PDR is calculated at the PDCP layer.

$$PDR = \frac{Data\ Received_{PDCP}}{Data\ Sent_{PDCP}}$$

At the PDCP layer we can map the new incoming packets and also the retransmitted ones by TCP. The RLC layer is enabled with AM mode then when a packet is transmitted, but not yet acknowledged, it is stored in the RLC AM retransmission buffer. Later, if a handover is performed, the RLC AM retransmission buffer is forwarded to the target eNB (RAT) and retransmitted. Consequently, if at the first time it was received successfully, it is redundantly transmitted.

The PDR results in this work are represented as the average value of the total volume transmitted/received at the PDCP layer per each simulation. The impact of varying the parameters of latency and buffer size among CCAs on the PDR when using dual connectivity is shown in Figure 5.4.5.

When the application rate is set at 1Gbps (Figure 5.4.5(a)), the impact in the PDR stays close to value of 1 for most of the cases. At 2ms latency with 1Mbytes buffer size, protocols such as NewReno, Hybla, HTCP, Illinois and Scalable experience 3% of PDR degradation. From the simulations, we observed that the protocols fell into constant RTO's event.

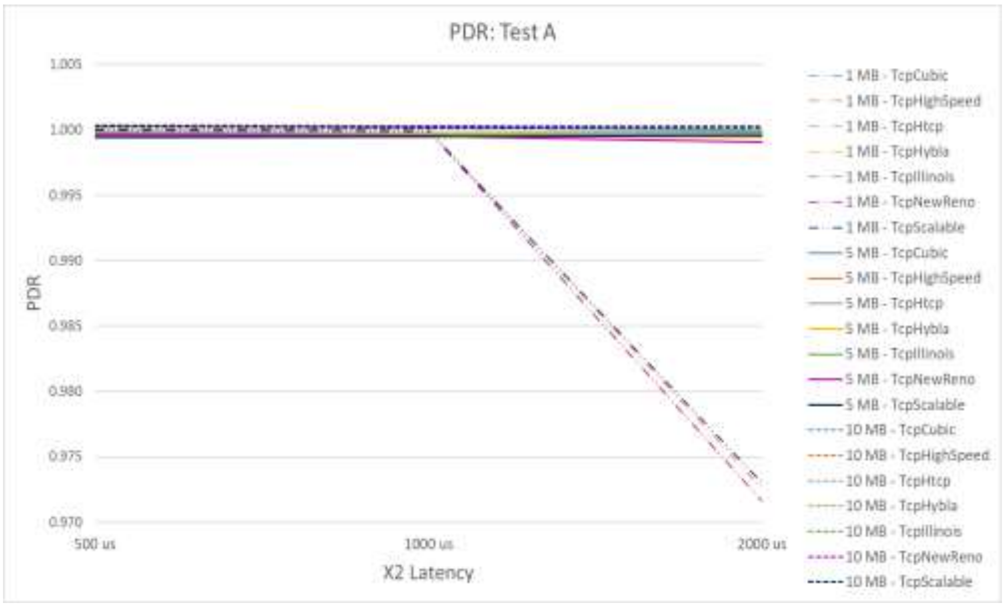
In test B, the PDR remains close to 1 but in case where the value undergoes 1 some protocols presents persistent RTO, like NewReno, Hybla, HTCP and Illinois.

In test C, the PDR is affected by the channel condition. What we can observe from the Figure 5.4.5(c) is that the PDR decreases by 2% is case of 10 MBbytes buffer size. Another observation found is that it starts to appear some trade-off between latency and buffering, when the latency is set at 1 ms, with 10 MB for most of the congestion control algorithm, for protocol, such as Scalable, HTCP and NewReno.

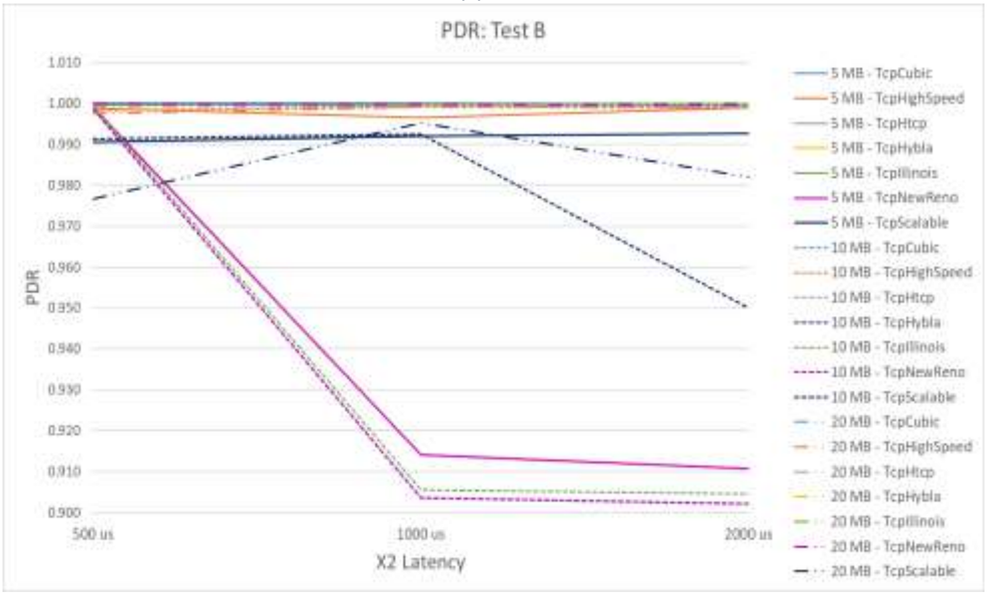
In test D most of the protocols decay in performance at 1 and 2 ms latency. In this simulation it has been observed that in NewReno, Hybla, HTCP and Illinois constant RTO events happened.

In case of NewReno and Hybla the performance rate is the lowest among the other protocols, because when a packet is lost the impact in the overall ratio is a bit higher. In case Scalable, even though it shows a stable performance rate we notice that its $cwnd$ size ramps up till eventually a congestion event happens.

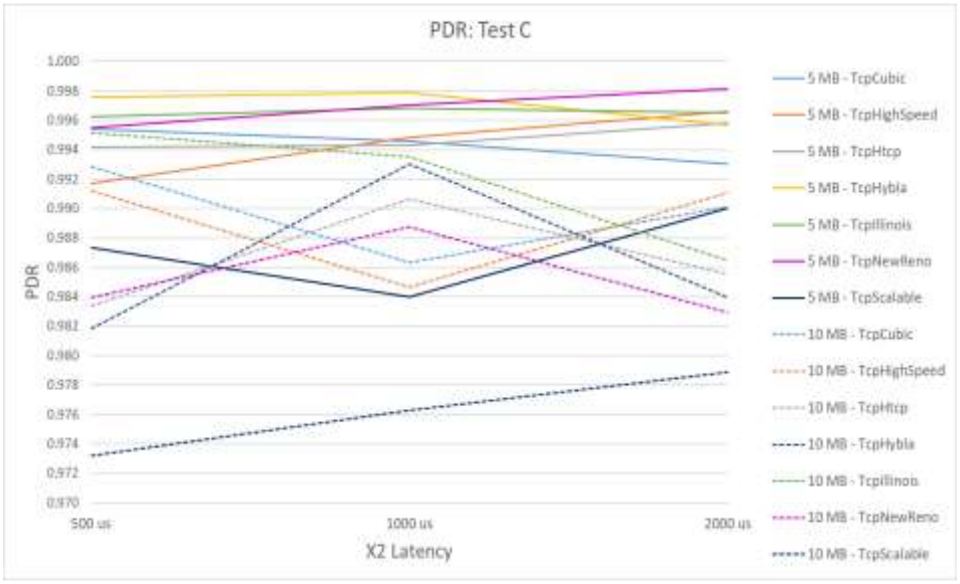
Considering the variant X2 latency and buffer size, we can see that Cubic, Illinois and Highspeed CCA show the most hopeful solutions.



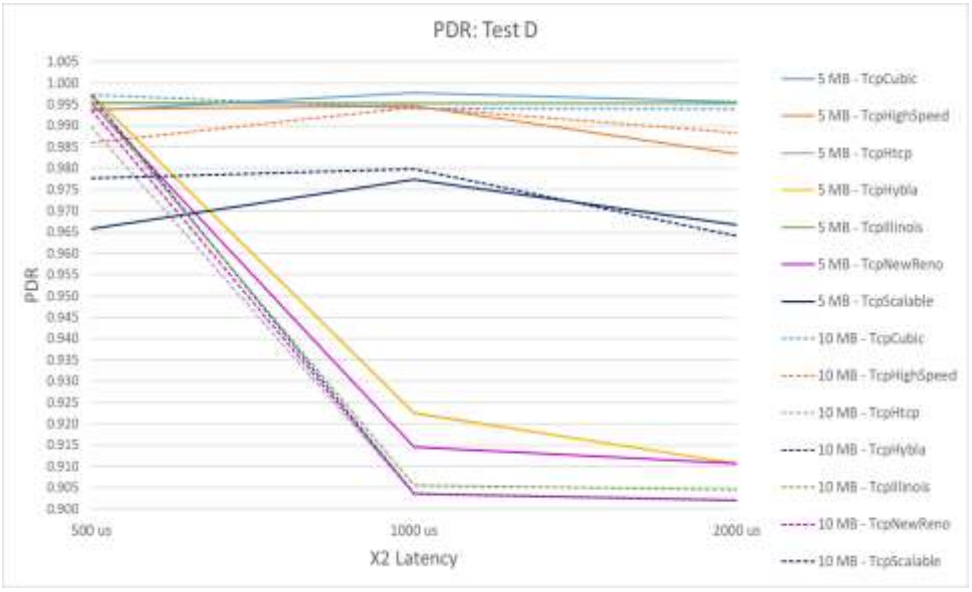
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.4.5: Comparison of the PDR at the PDCP layer for the different scenarios.

5.4.6. Bytes in-flight

The Bytes in-flight correspond to the number of unacknowledged packets sent, by the application, which are travelling along the system till the user receives them and sends acknowledgment packets to the TCP remote host.

The Bytes in-flight are linked with the instantaneous RTT (25 milliseconds) times the Throughput (1Gbps), which corresponds to 3 Mbytes approximately and 9 Mbytes for 3Gbps. This means that by incrementing X2 Latency or the system rate, the number of Bytes in-flight will grow.

The Figure 5.4.6 shows the average Bytes in-flight for tests A, B C and D, considering 5MB and 10 MB values at the RLC. The result from tests B and D show that 90 % of the mean bytes in-flight are below 9 MB in test B and 80 % in test D for Hybla, NewReno, Illinois and Cubic.

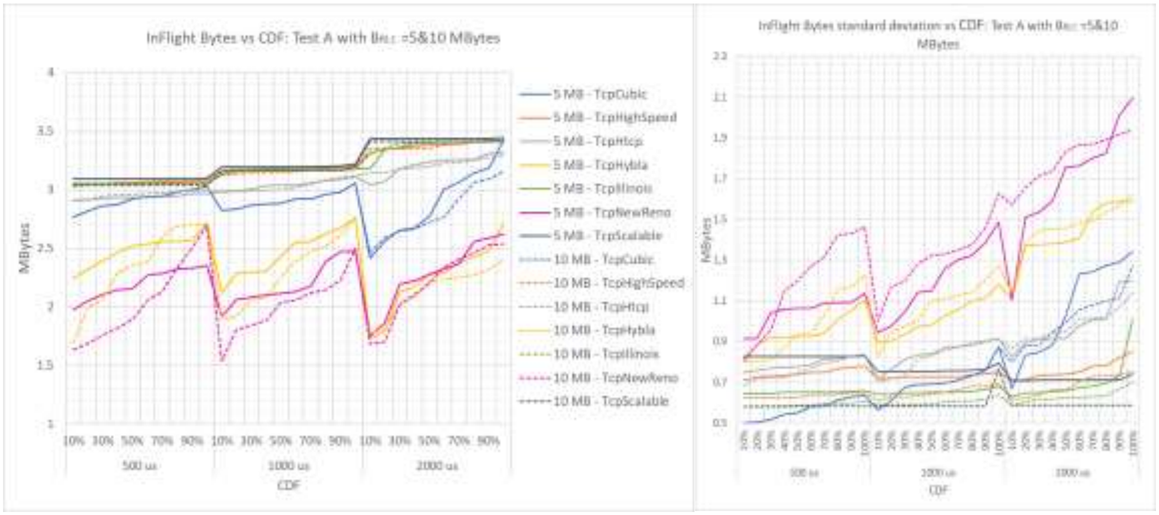
The results from tests A and C show that the number of bytes in-flight are around 3 Mbytes, which was previously estimated, less for New Reno and Hybla, where the Bytes in-flight are close to 2MB and their standard deviation too (in test A the standard deviation is approximately between the 30 % - 40% for Hybla at 2000us and New Reno at 2000us). Originated during the first handover, in both cases the packets are arriving in disorder, this incident triggers a congestion event and the congestion window is reduced to half of the number of Bytes in-flight, for both congestion algorithms. After that, the increment of the congestion window of New Reno for is very slow since it increases approximately by half packet per round trip plus handover events might keep halving the window, turning nearly impossible to return to the target rate. On the other hand, Hybla congestion avoidance algorithm is based on New Reno's algorithm but reduces the RTT dependency, which is mainly caused by secondary cell handovers.

In test A, we can see how the number of bytes inflight increase with the latency. Protocols, such as NewReno and Hybla, present a lower traffic and higher variation.

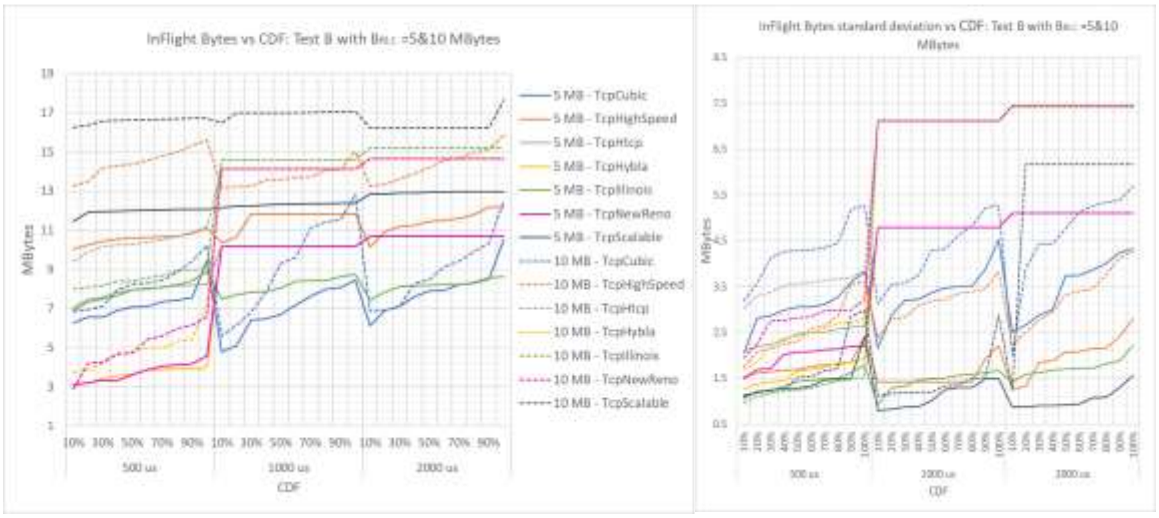
In test B, we can see the number of Bytes in-flight grow because of the higher rate requested. Something we notice is that, in case of Illinois, at 5 MB shows steady performance as we variate the X2 latency. The same goes for its standard deviation. Some straight lines appear in the graphic. Those represent the cases where constant RTO events occur.

In test C and D, we can see that the number of bytes inflight are increased because of the degradation in the channel forced by delays and retransmissions.

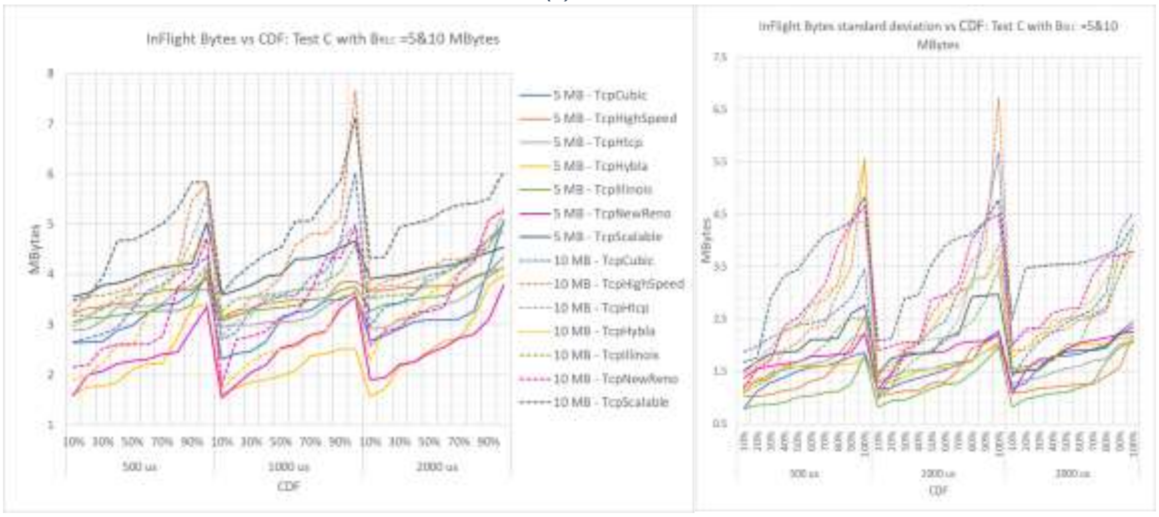
Considering these results in terms of goodput and number of bytes in-flight standard deviation, can conclude that Illinois is the most promising solution.



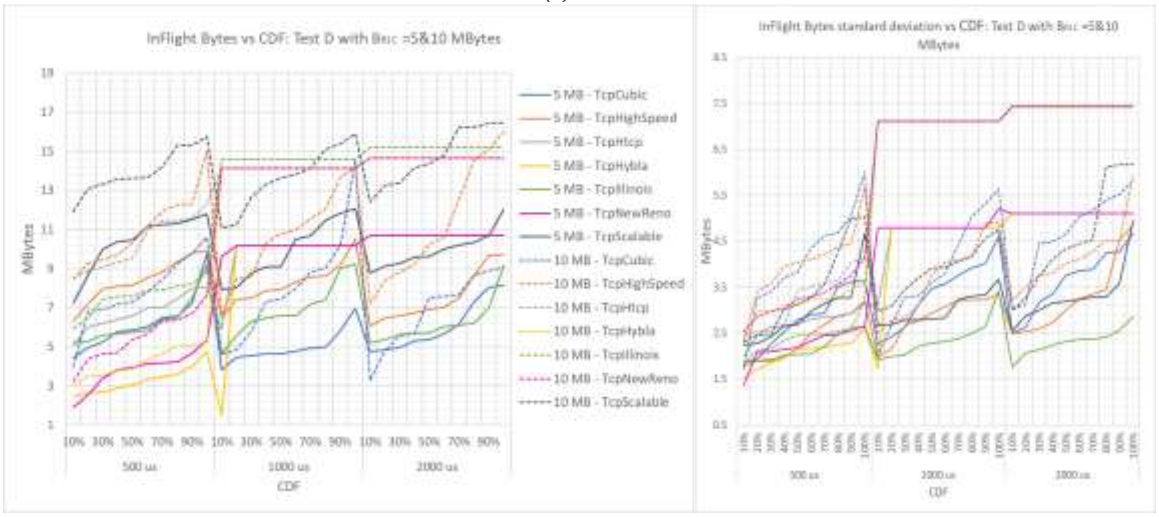
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.4.6: Overall bytes in flight for every protocol. It considers 5 MB and 10MB RLC buffer.

5.4.7. Round-trip-time (RTT)

The round-trip time (RTT) is an important metric in determining the behaviour of a TCP connection, it is defined as the time from when a packet is sent to when the time an ACK for that packet is received. The round-trip time calculation is based on the timestamp option (allowing to time every segment including retransmissions). From Figure 5.3.4, Figure 5.3.8, Figure 5.3.11, Figure 5.3.14, Figure 5.3.18, Figure 5.3.21 and Figure 5.3.24 we can observe that the RTT is incremented as we raise the X2 latency.

The X2 latencies were set up at 0.5, 1 and 2 ms for the simulations, which contributes proportionally to RTT over the system. This relation impacts in the TCP performance goodput by a decrement in the data rate. Moreover, in scenario 2 we can see that due to the LOS/NLOS environment the RTT increases the fluctuations.

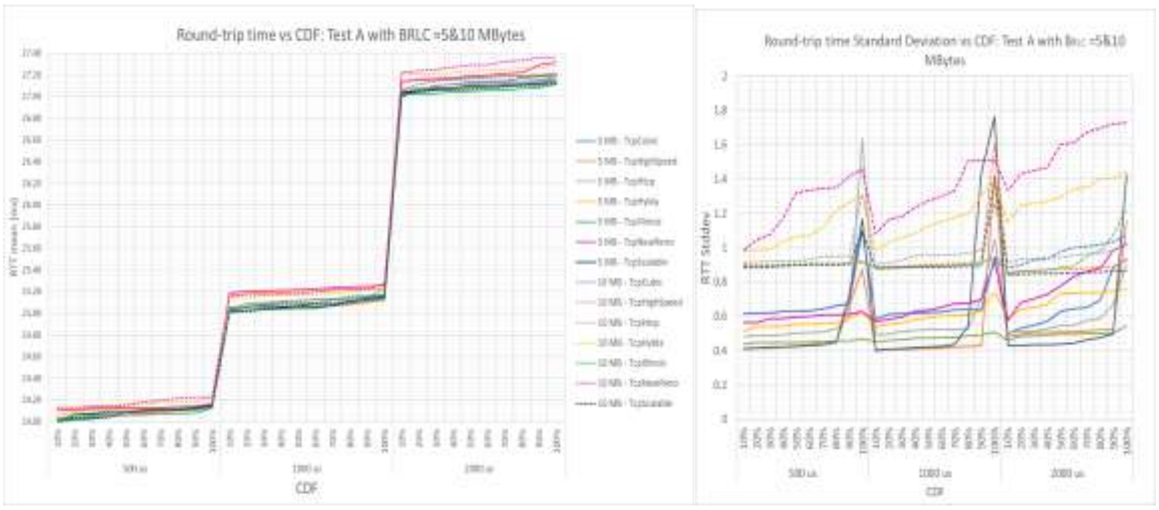
The standard deviation turns to be a helpful indicator to evaluate the different CCA over the configuration we defined at the begging of this chapter. The Table 5.5 shows the average and standard deviation of the round-trip time (RTT) with the Goodput per CCA varying the Buffer (5 and 10 Mbytes) and X2 latency (0.5,1,2 milliseconds). The average RTT observed among the different CCA was 24.09 microseconds ($B_{RLC} \ = 5 \, Mbytes$ and $D_{X2} = 500us$) with a standard deviation between +/- {0.45; 0.68} getting a goodput in the order of 900Mbps in simulations for 1Gbps target (in most of the CCA except NewReno and Hybla). However, when we recreate the scenario 2 under the same conditions, we can see how the standard deviation is up to 10 times the variability which we can see how it impacts the average goodput, 673Mbps in case of Illinois (best result).

The standard deviation can be used to evaluate the variability of RTTs within CCAs. Figure 5.4.7 shows the cumulative distribution function (CDF) of the standard deviation per set of configurations of the 4 scenarios proposed. In test A, 80% of the standard deviation is within 0.4 and 0.6 milliseconds for 5MBytes buffer with and average RTT around 24.1 milliseconds and 25.1 milliseconds for X2 latencies of 0.5 and 1 milliseconds respectively. However, we can see for 2 milliseconds latency, the standard deviation increases for protocols like NewReno and Hybla. In case of the buffer set up at 10 Mbytes, 90% of the standard deviation is below 1 millisecond for most of the CCAs except for NewReno and Hybla. In addition, the CDF of the average RTT shows how the RTT increments as the X2 latency grows.

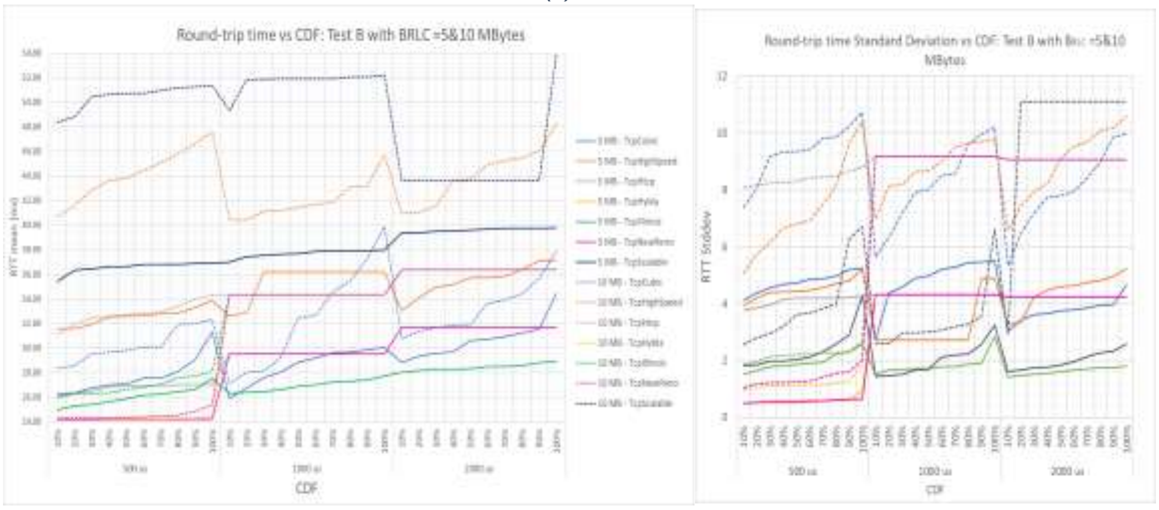
In test B, we can observe an increment on the RTT. Most of the CCA experience a low standard deviation (around 2 ms) with 75% experiencing RTT below 28 ms, for 0.5 ms for the case of Illinois and Cubic. Protocols such as HighSpeed and Scalable, present high degree of delay at 10 Mbytes buffer, with high standard deviation.

In scenario 2, we observe that 60% of the RTT is below 30 ms with a standard deviation below 10 ms, which correspond to 5 Mbytes buffer size, in test C.

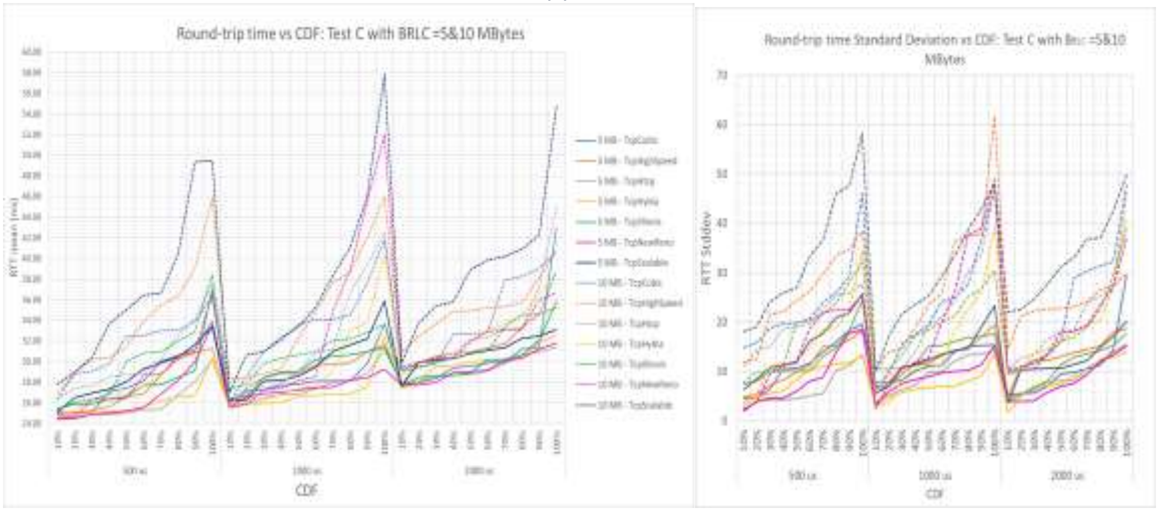
In test D we can observe that 60% of the simulations present RTT mean below 32 ms with the fluctuation under 10 ms.



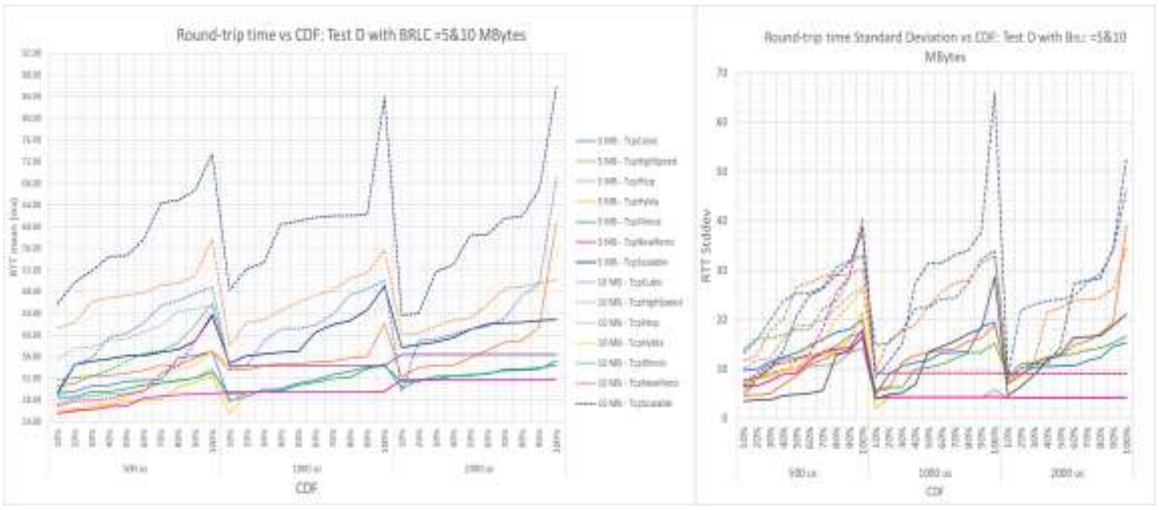
(a) Test A



(b) Test B



(c) Test C



(d) Test D

Figure 5.4.7: (left) Average RTT (milliseconds) per simulation vs CDF. (right) RTT standard deviation per simulation vs CDF over the different X2 latencies. The solid lines represent the 5Mbytes buffer and the dashed lines represent the 10 Mbytes buffer, with the same colour per CCA variant.

The block error rate (BLER) is a measurement that calculates the ratio of the number of erroneous blocks transmitted. The figure below maps the block error-rate (BLER) vs the average RTT for the different latency and buffer configuration. We can observe for the different configuration that the BLER remains between 10^{-1} and 10^{-2} for the different rate tests over the varying latency.

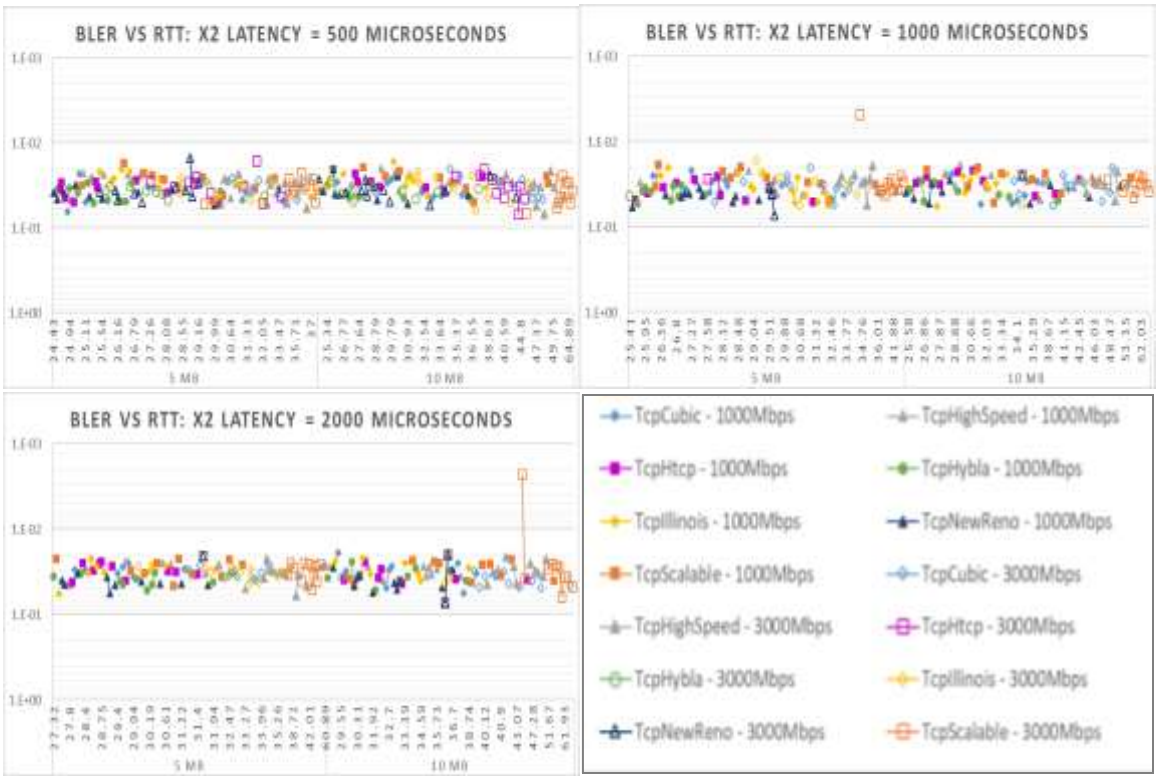


Figure 5.4.8: Round-trip time average versus average BLER

5.4.8. Goodput and Its Variation

In order to provide certain level of confidence every configuration was performed 10 times. The Figure 5.4.9 to Figure 5.4.15 show the average goodput and the standard error of the mean for 1Gbps and 3Gbps rates for each TCP algorithm and fixed RLC buffer along varying X2 latencies. The dashed lines and plain lines correspond to 1Gbps and 3Gbps tests respectively.

The used metric for variation was the standard error of the mean (SEM) which measures how far the sample mean of the data is likely to be from the true population mean.

$$SEM = \frac{\sigma}{\sqrt{N}}$$

Where: σ is the standard deviation ($\sigma = \sqrt{\frac{\sum_{i=0}^{N}(x_i - \bar{x})^2}{N-1}}$) and N is the sample size.

We investigate 7 CC algorithm over mmWave for DC. Different protocols were tested to TCP Performance analysis over mmWave: NewReno, HighSpeed, HTCP, Scalable, Cubic, Illinois and Hybla. As mentioned earlier, the main cause of throughput degradation is handovers. The X2 latency between mmWave nodes, buffer capacity, congestion algorithm reactivity plays an important part for the target achievability.

It can be seen that Cubic outperforms others in terms of proximity to the target and the samples are more around the true population, for 1000Mbps test. Other examples are HighSpeed, Illinois, HTCP and Scalable which show a low variation but in the case of large latency with small buffering (1MB) it shows RTO events permanently.

In case of 3000 Mbps test, the mean Goodput are more spread out which means that it becomes more likely that any given mean is an inaccurate representation of the true population mean in case of HighSpeed, Hybla, Cubic and Scalable. Regarding Illinois and NewReno variation, the graph show stability when RTO event occurs.

In scenario 2, we can see that, protocols, such as NewReno, Hybla, Illinois and HTCP experience constant RTO events at 1000ms and 2000 ms with 10 MB buffer at 3Gbps. HighSpeed, Cubic and Illinois withstand the system conditions.

Notice that when the Goodput is close to zero the SEM values is zero, this means that in those cases the simulation fell in constant RTO events.

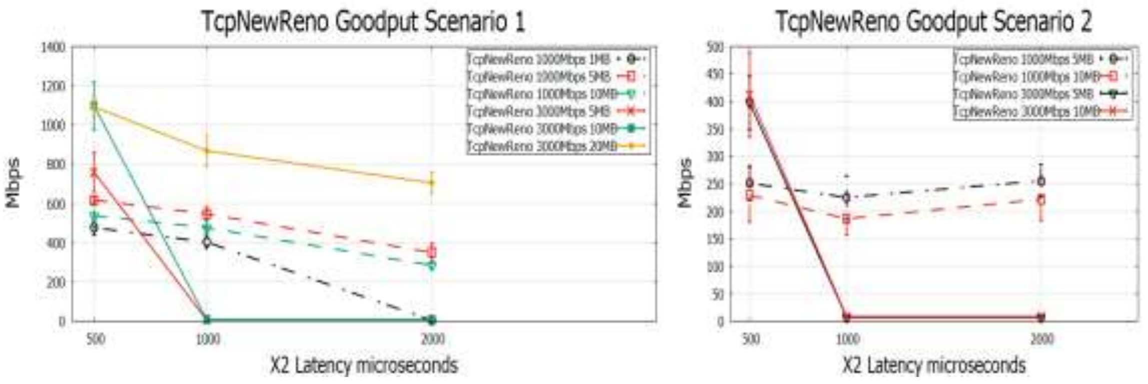


Figure 5.4.9: TCP NewReno Average Goodput and SEM. (left) scenario 1, (right) scenario 2

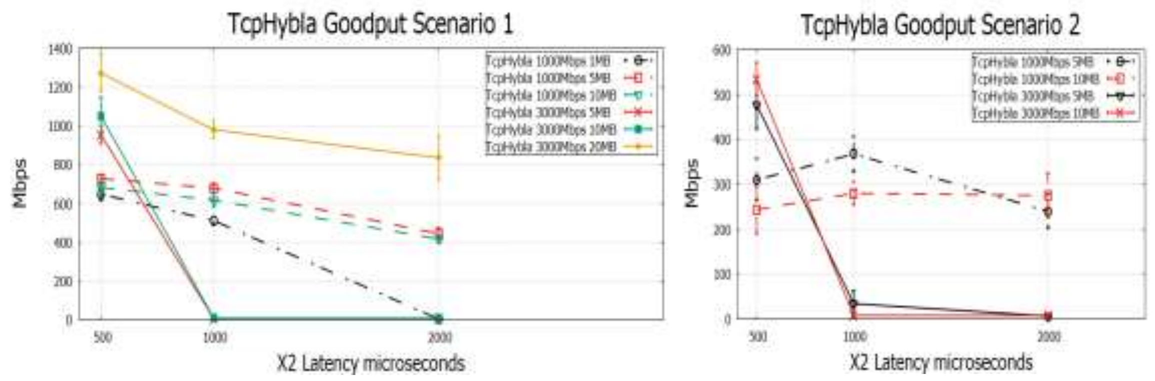


Figure 5.4.10: TCP Hybla Average Goodput and SEM. (left) scenario 1, (right) scenario 2

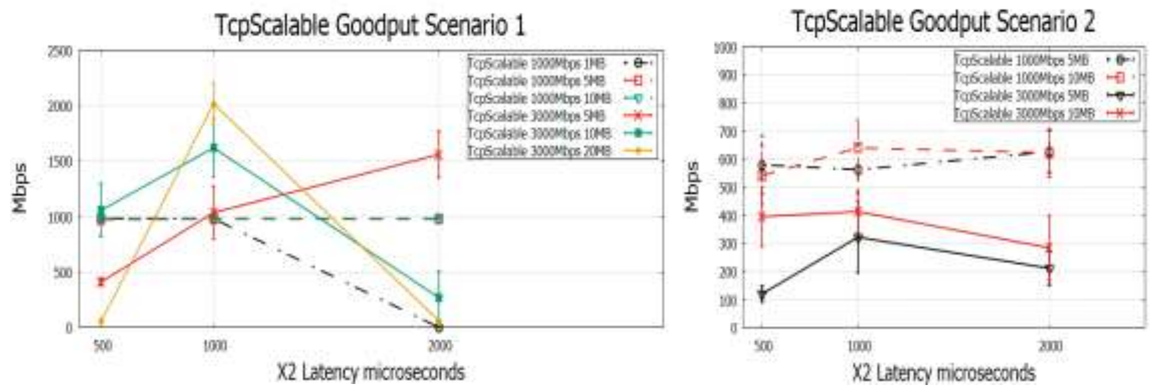


Figure 5.4.11: TCP Scalable Average Goodput and SEM. (left) scenario 1, (right) scenario 2

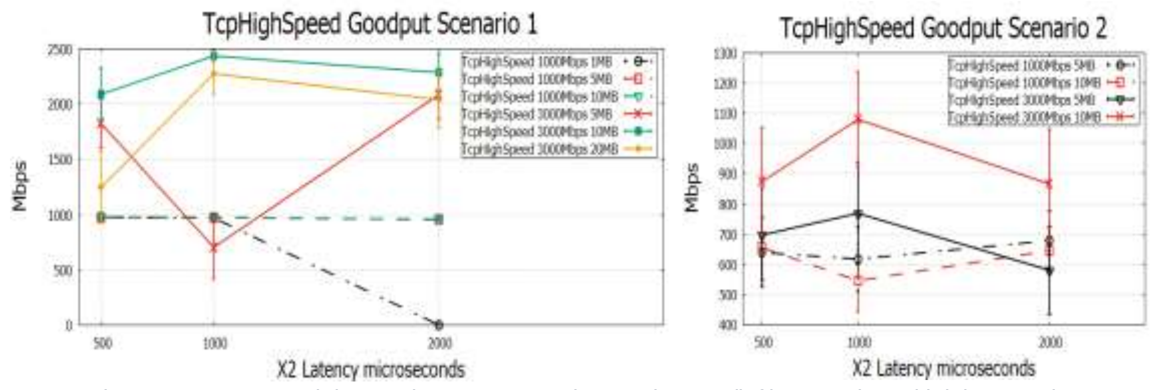


Figure 5.4.12: TCP HighSpeed Average Goodput and SEM. (left) scenario 1, (right) scenario 2

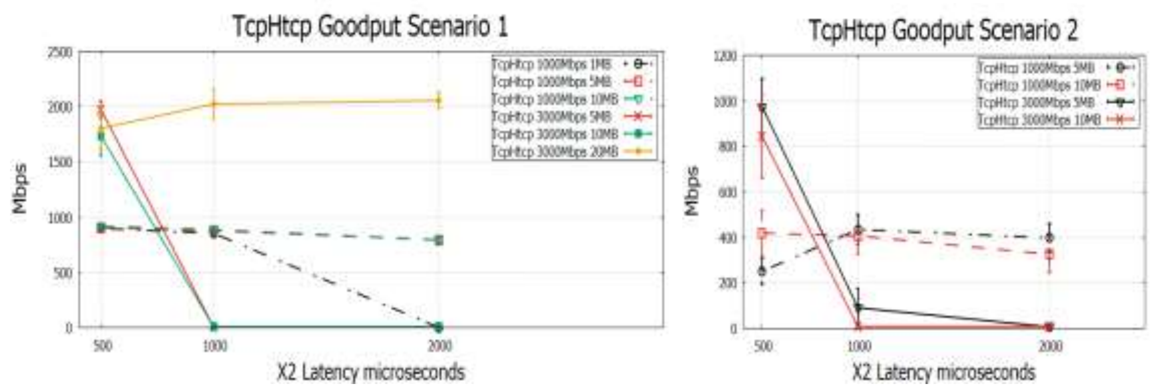


Figure 5.4.13: HTCP Average Goodput and SEM. (left) scenario 1, (right) scenario 2

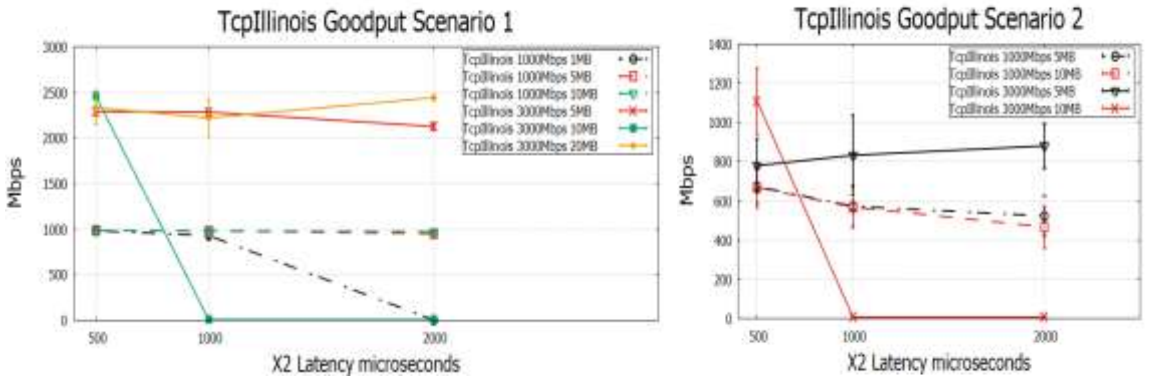


Figure 5.4.14: TCP Illinois Average Goodput and SEM. (left) scenario 1, (right) scenario 2

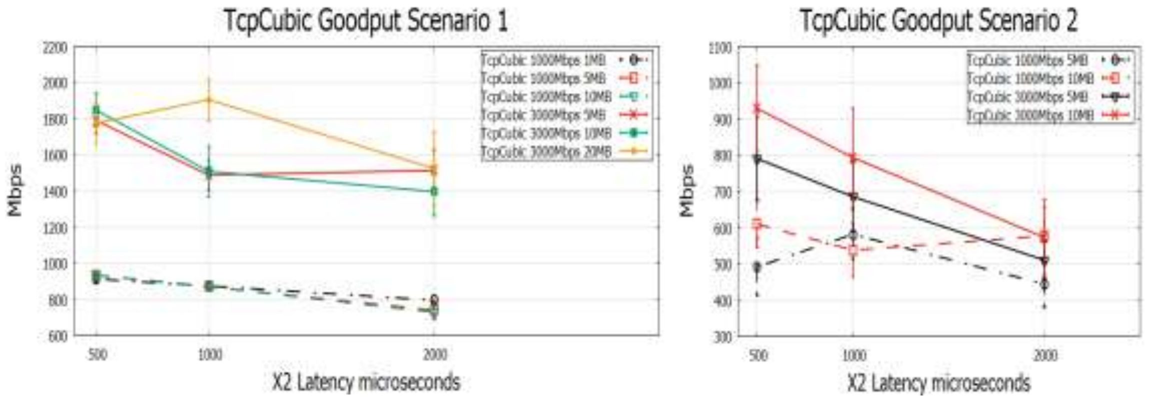


Figure 5.4.15: TCP Cubic Average Goodput and SEM. (left) scenario 1, (right) scenario 2.

6. Budget

Regarding the financial costs for the project, there were not much additional expenses:

The hardware used were two desktop computers and extra storage:

- Desktop 1 used for the study and development phases (HP Compaq 8100 Elite SFF Core i5-650 CPU 3.20GHz, 4GB memory, 250GB hard disk space) provided by UPC.
- Desktop 2 used for the simulation campaign due to the high computational process and number of simulations needed (intel Core X299 UD4 Pro, 200GB hard disk space) provided by UPC.
- Hard Drive with 3 Terabytes capacity. Cost 100 euros.

The software that were used were free and open source (Ubuntu 16.04.4 LTS, ns-3, python and Gnuplot, etc.). In addition, a backup storage was needed so I purchased a Dropbox account for 8 months (20 euros per month). Cost 160 euros.

About the man-hours for roughly 9 months or 38 weeks that were spent on the project from February 2019 to October of 2019. So, assuming an average of 40 hours done per week, the total labour devoted on the project was 1520 man-hours.

7. Conclusions and future development

In this work, we evaluate the performance of 7 CCAs over Dual Connectivity on ns-3 simulator by measuring performance metrics such as Goodput, Bytes in-flight, RTT and PDR at PDCP layer. From this test evaluation, the following specific conclusions can be drawn:

Goodput values for the seven congestion control algorithms over the four-test model defined are comparable. The values between scenarios 1 and 2 are almost half and in some cases the CCA falls in to constant RTO events that with a small $cwnd$ strategy makes almost impossible to overcome that situation, e.g. NewReno, Hybla. On the other hand, CCAs like Scalable and HTCP show that a $cwnd$ increment without any feedback turn dangerous for the network since these algorithms present the highest $cwnd$ steeping increment. Finally, Cubic, Highspeed and Illinois present a steady performance among the different scenarios and set of parameters configured.

PDR values were calculated at the PDCP layer and compared CCAs along the increment of the latency, which show that some protocols present a stable traffic success at 1000 ms X2 latency at high traffic load in LOS scenarios (test B). In scenario 2 the most resilient CCA algorithm are Scalable, Illinois, and Cubic for both 5MBand 10 MB buffer size.

RTT values all follow a similar pattern for the 4-test models and CCA. However, algorithms like Hybla and NewReno show a flat performance since after some congestion events the $cwnd$ is reduced that much that it barely increases, this means that in terms of RTT values this would be almost perfect. Cubic, Highspeed and Illinois showed acceptable RTT values and a lower jitter value as well.

Bytes in-flight values grew proportional to the traffic load so we can see that Illinois shows more stability compare with the other CCA analysed considering that in terms of goodput it was acceptable.

Finally, in terms of level of RTT variation, CCAs like Illinois, HTCP, Scalable, Hybla and NewReno showed better values. Cubic does too but it seems to be less accurate for low X2 latencies.

Bibliography

[1] 3GPP TR 36.842: "Study on Small Cell enhancements for E-UTRA and E-UTRAN; Higher layer aspects," V12.0.0, Dec. 2013.

[2] S. C. Jha, K. Sivanesan, R. Vannithamby, A. T. Koc, "Dual connectivity in LTE small cell networks", Proc. IEEE Globecom Workshops (GC Wkshps), pp. 1205-1210, Dec. 2014.

[3] R. Antonioli, G. Parente, Carlos F. M. e Silva, E. B. Rodrigues, T. F. Maciel, Fco. R. P. Cavalcanti, "Dual Connectivity for LTE-NR Cellular Networks", XXXV SIMPOSIO BRASILEIRO DE TELECOMUNICACIONES E PROCESSAMENTO DE SINAIS, 2017

[4] [Online]. Available: https://www.nsnam.org/docs/release/3.28/models/html/index.html.

[5] [Online]. Available: https://www.nsnam.org/docs/release/3.28/tutorial/ns-3-tutorial.pdf.

[6] [Online]. Available: https://www.nsnam.org/docs/release/3.28/manual/ns-3-manual.pdf

[7] https://www.nsnam.org/wiki/Installation

[8] MmWave module [Online]. Available: www.github.com/nyuwireless-unipd/ns3-mmwave/tree/new-handover.

[9] LTE-EPC Network Simulator. [Online]. Available: www.iptechwiki.cttc.es/LTE-EPC_Network_Simulator_(LENA).

[10] 3GPP, Sophia Antipolis, France, Rep. TR 38.900 "Study on channel model for frequency spectrum above 6 GHz," V14.2.0, 2017.

[11] M. Polese, M. Mezzavilla, and M. Zorzi, "Performance comparison of dual connectivity and hard handover for LTE-5G tight integration," in Proc. 9th EAI Int. Conf. Simulat. Tools Techn. (SIMUTOOLS), Prague, Czech Republic, 2016, pp. 118–123. [Online]. Available: http://dl.acm.org/citation.cfm?id=3021426.3021445

[12] R. Antonioli, G. Parente, Carlos F. M. e Silva, E. B. Rodrigues, T. F. Maciel, Fco. R. P. Cavalcanti, "Dual Connectivity for LTE-NR Cellular Networks", XXXV SIMPOSIO BRASILEIRO DE TELECOMUNICACIONES E PROCESSAMENTO DE SINAIS, 2017

[13] M. Polese, M. Giordani, M. Mezzavilla, S. Rangan, and M. Zorzi, "Improved handover through dual connectivity in 5G mmWave mobile networks," IEEE J. Sel. Areas Commun., vol. 35, no. 9, pp. 2069–2084, Sep. 2017.

[14] I. D. Silva, G. Mildh, J. Rune, P. Wallentin, J. Vikberg, P. Schliwa-Bertling,and R. Fan, "Tight Integration of New 5G Air Interface and LTE to Fulfill 5G Requirements," in 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), May 2015, pp. 1-5.

[15] 3GPP Technical Specification 36.300, "Evolved Universal Terrestrial RadioAccess (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description." [Online]. Available: www.3gpp.org

[16] S. Ha, I. Rhee, and L. Xu, \CUBIC: a new TCP-friendly high-speed TCP variant," ACM SIGOPS Operating Systems Review, vol. 42, no. 5, pp. 64{74, 2008.

[17] S. Ha and I. Rhee. Taming the elephants: New tcp slow start. Computer Networks, 55(9):2092–2110, 2011.

[18] H. Wang, C. Rosa, and K. Pedersen, “Dual connectivity for LTEadvanced heterogeneous networks,” Wireless Networks, pp. 1–14, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11276-015-1037-6

[19] T. Wigren, K. Lau, R. Delgado, and R. H. Middleton, “Delay skew packet flow control in wireless systems with dual connectivity,” IEEE Trans. Veh. Technol., to be published.

[20] M. S. Pan, T. M. Lin, C. Y. Chiu, and C. Y. Wang, “Downlink Traffic Scheduling for LTE-A Small Cell Networks with Dual Connectivity Enhancement,” IEEE Communications Letters, vol. 20, no. 4, pp. 796– 799, April 2016.

[21] S. Xu and Y. Fu, "Resource allocation algorithm for maximizing network utility in LTE network with dual connectivity," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 600-606.

[22] P. Legg, P. Fotiadis and P. Soldati, "Load Balancing and Aggregation Algorithms for LTE Dual Connectivity," 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, 2016, pp. 1-5. doi: 10.1109/VTCSpring.2016.7504321.

[23] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. The NewReno Modification to TCP’s Fast Recovery Algorithm. RFC 6582 (Standards Track), 2012.

[24] Wooseong Kim, “Dual Connectivity in Heterogeneous Small Cell Networks with mmWave Backhauls,” Mobile Information Systems, vol. 2016, Article ID 3983467, 14 pages, 2016. https://doi.org/10.1155/2016/3983467.

[25] Modarresi, Amir & Gangadhar, Siddharth & Nguyen, Anh & Sterbenz, James. (2016). H-TCP Implementation in ns-3 (extended version). 10.13140/RG.2.2.19765.47845.

[26] D. Leith, R. Shorten, and Y.Li. H-TCP: A framework for congestion control in high-speed and long-distance networks. HI Technical Report, August 2005. Available at http://www.hamilton.ie/net/htcp/.

[27] NGMN Alliance, NGMN 5G White Paper, Feb. 2015; [Online] Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf , Accessed 7 Feb 2018.

[28] 3GPP TS 23.501 “System Architecture for the 5G System; Stage 2”. V15.2.0 Release 15, 2018.

[29] 3GPP TR 38.470, “Transport network support of IMT-2020/5G”, 2018-02.

[30] 3GPP TS 38.300: "NR; Overall description; Stage-2", version 15.3.1 Release 15.

[31] Manoharan Ramalingam “Differences between 5G NR and LTE Layer-2 protocols” [Online] Available: https://www.linkedin.com/pulse/differences-between-5g-nr-lte-layer-2-protocols-manoharan-ramalingam/ [Accessed: 19 July 2018].

[32] 3GPP TS 38.401: "NG-RAN; Architecture description",version 15.3.0 Release 15

[33] 3GPP TS 38.300: "NR; Overall description; Stage-2", version 15.3.1 Release 15

Appendices

A. 5G System

System Architecture and services

The 5G system architecture is service-based which means that the architectural elements are composed by different Network Functions (NF). The NFs offer their services via interfaces of a common framework to any network functions that are permitted to use these services. The interactions between network functions is represented in two ways, [28]:

- service-based representation: The network functions within the Control Plane enables other authorized network functions to access their services.
- Reference point representation: it shows the interaction between the NF services for providing system level functionality and to show inter-PLMN interconnection across various network functions.

The 5G architecture is composed of a converged core network (5GC) with a common Access network (AN) and Core network interface which integrates different Access types e.g. standalone and non-standalone access (in Release 15 these are defined the 3GPP NG-RAN and the 3GPP defined untrusted WLAN access. The different network entities are connected by TCP/IP, which supports diff-serv QoS), Figure A.1.

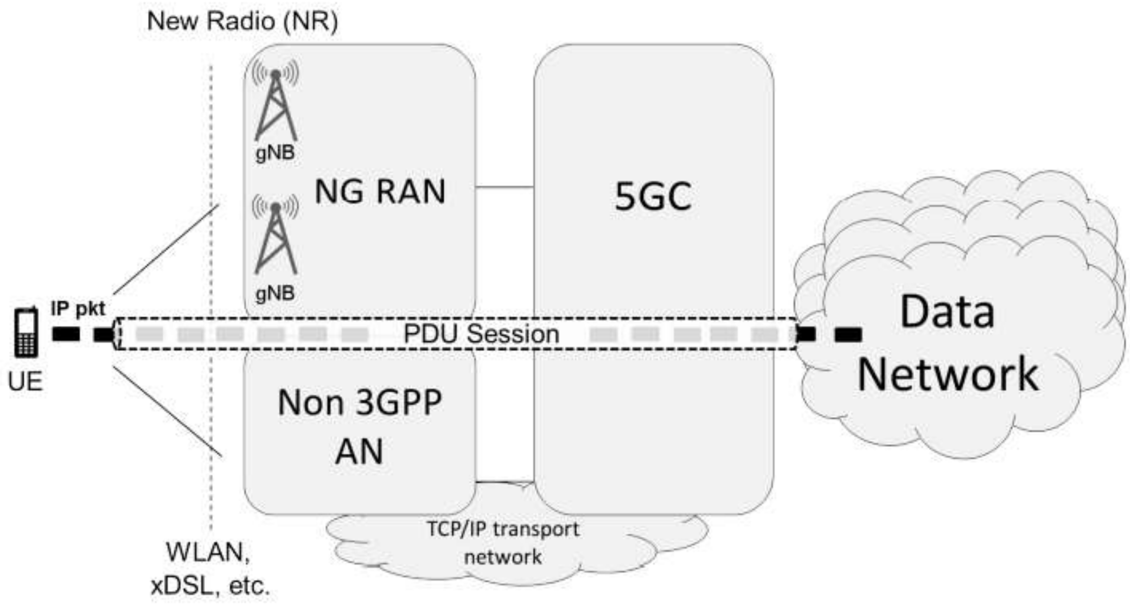


Figure A.1: 5G Network Architecture.

The 5G connectivity service is named PDU Session. The PDU session is a sequence of NG tunnels in 5GC, and of one or more radio bearers at the radio interface, Figure A.2. Upon the UE request, the connect to its control functions and to the external data network is stablished. The PDU session not only transport user plain IP packets, but also ethernet or not structured frames i.e. it allows L2 communications.

Regarding the QoS model, a new model is defined based on the new concept of QoS flow where a single flow is considered as the finest granularity of QoS differentiation in a PDU session. This means that the User plane traffic belonging to the same QoS flow receives the same traffic forwarding treatment. At the radio interface the QoS flows are mapped to data radio bearers.

Figure A.3 shows the splits between the 5G functions, where the NG-RAN takes care of establishing, maintaining and releasing the parts of the PDU sessions that cross the radio interface and the 5GC functions manage the remaining parts of the PDU sessions and take care of all the other processes not related to radio access.

The 5GC is aware of the service requirements and of QoS control for each QoS flow. In the downlink, the UPF maps and mark the IP packets to QoS flows, QoS Flow Identifier (QFI), which is useful to assist UL QoS handling. At the NG-RAN, the gNB maps the QoS flows to data radio bearers. This is done at the new SDAP layer of the radio interface.

In the uplink, the mapping between QoS flows and data radio bearers is done at the UE and it can be performed in two different ways: Explicit mapping (configured by the network using RRC signalling) and Reflective QoS mapping (the UE knows which QoS flow and DRB ae mapped to which IP packets by observing the DL packets).

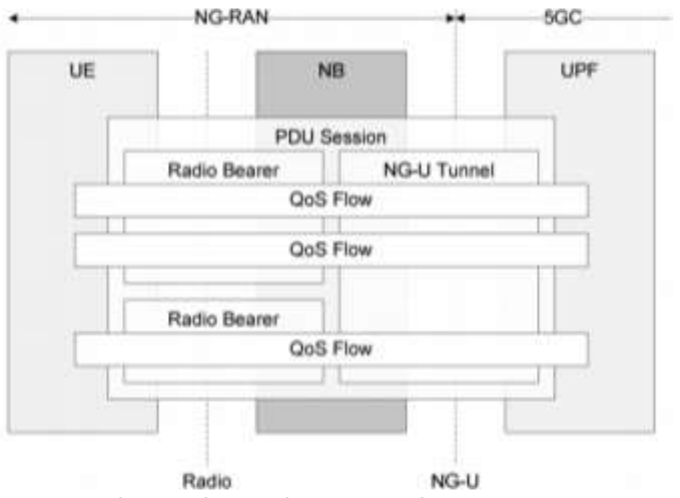


Figure A.2: PDU Sessions and QoS Flows: User Plane. Source 3GPP TS 38.300, [30]

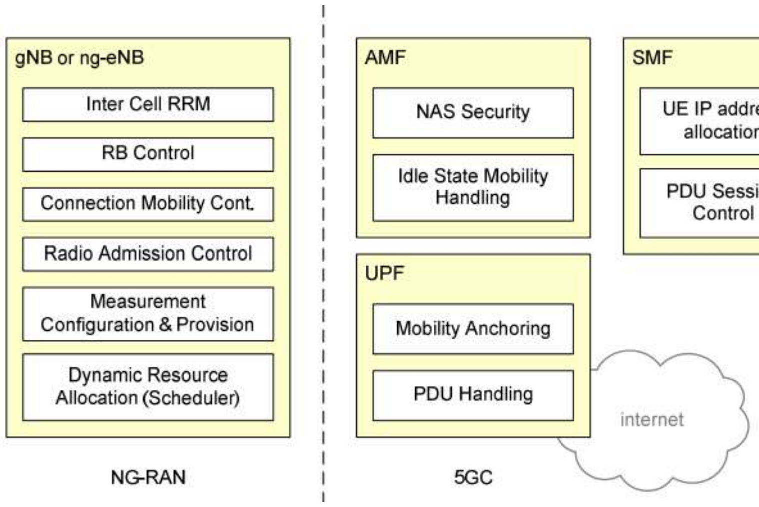


Figure A.3: Functional Split Between NG-RAN and 5GC. Source 3GPP TS 38.300, [30]

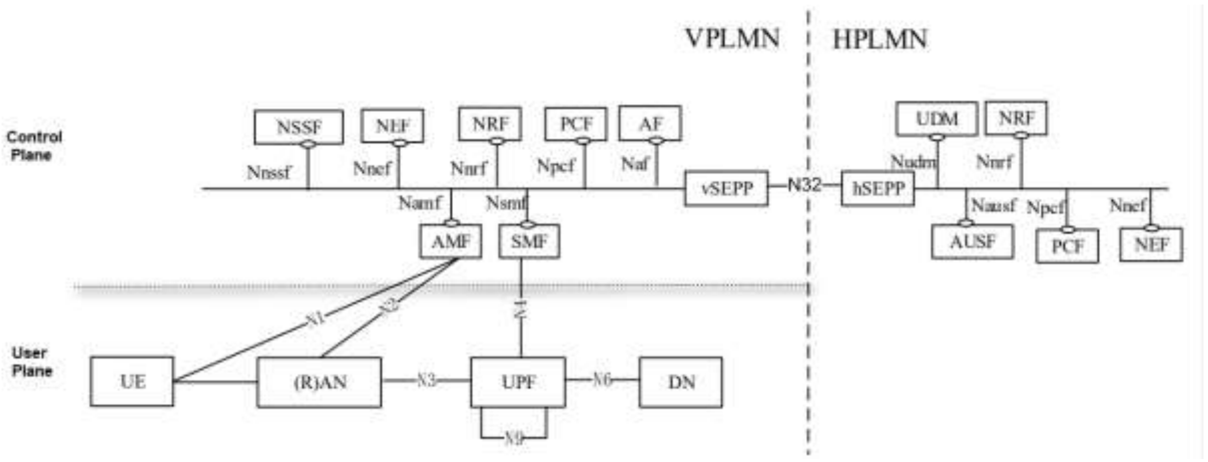


Figure A.4: Roaming 5G System architecture in service-based interface representation (source 3GPP TS 23.501, [28])

5G Core Network (5GC)

The separated network node functions allow to define the 5G architecture in terms of network functions, indeed the 5G core network presents a service-based architecture. The interaction between network functions (procedures) are defined as services.

Figure A.4 shows one of the service-based architecture defined in [28]. In a local breakout scenario, the roaming UE interfaces the Data Network (DN) in the visited network (VPLMN) and the home network (HPLMN) enables it with subscription information from Unified Data Management (UDM), Authentication Server Function (AUSF) and UE specific policies from Policy Control Function (PCF), [28].

In the 5G control plane the NFs are exposed as service-based interfaces i.e. the connection is by a network bus rather than by point-to-pint links. The interface is an API that any other entity could use e.g. to discover and enable communication with other NFs the NF queries an NRF. Finally, in the 5G user plane the entities are still interface by point-to-point interfaces, Figure A.4. The **Error! Reference source not found.** describes t he main roles of network functions (NF) stated in [28].

Authentication Server Function (AUSF)	The AUSF provides UE authentication service.
Access and Mobility Management Function (AMF)	The AMF is a control plane function in charge of handling the control signalling between the UE and the 5G Core Network (5GC) e.g. operator services, Internet access or 3rd party services.
Unstructured Data Storage Function (UDSF)	Storage and recovery of information as unstructured information.
Network Exposure Function (NEF)	It is used by NFs to display capabilities and events to other NFs. NEF receives information from other NFs and stores it in the UDR. The NEF can access to the stored information and re-expose it to other NFs and application functions.

Network Repository Function (NRF)	The NRF allows every NF to discover the services offered by other NF Repository with the profile of the available NF instances (id, PLMN ID, network slice identifiers, capacity information, etc.) and their supported services.
Network Slice Selection Function (NSSF)	Supports the selection of the Network Slice instance(s) serving a UE
Policy Control Function (PCF)	Provides policy rules to control plane functions and a unified policy framework to govern the network actions.
Session Management Function (SMF)	The SMF is a control plane function in charge of session management, UE IP address location and management, control of policy enforcement and QoS, etc.
Unified Data Management (UDM)	User identification handling, subscription management, access authorization based on subscription data (e.g. roaming restrictions), generation of authentication credentials.
Unified Data Repository (UDR)	Store subscription data.
User Plane Function (UPF)	It provides the user plane and it is managed following a model of control and user plane separation. It acts as a gateway between the RAN and the external Data Network (DN).
Application Function (AF)	It resembles an application server that require dynamic policy and/or charging control (e.g. it interacts with these applications and provides policy requirements to the PCF).
Security Edge Protection Proxies (SEPP)	The SEPP protects the interactions between PLMNs

Table A.1: Network functions description. Source: TS 3GPP 123.501, [28]

NG-RAN

The NG-RAN is the network function that connects through a set of logical interfaces the User Equipment with the 5G Core. It can be composed by two types of nodes: the gNode B (gNB) that operates with the 5G New Radio (NR) technology and the next generation eNodeB (ng-eNB) that operates with the LTE technology.

The gNB as in LTE is responsible for all the radio-related functions associated to one or multiple cells supporting the new 5G NR functionalities such as RRM, routing, QoS flow management, etc. It may consist of a gNB-CU (central unit) and one or more gNB-DUs (distributed unit) interconnected via logical interface (F1),Figure A.5.

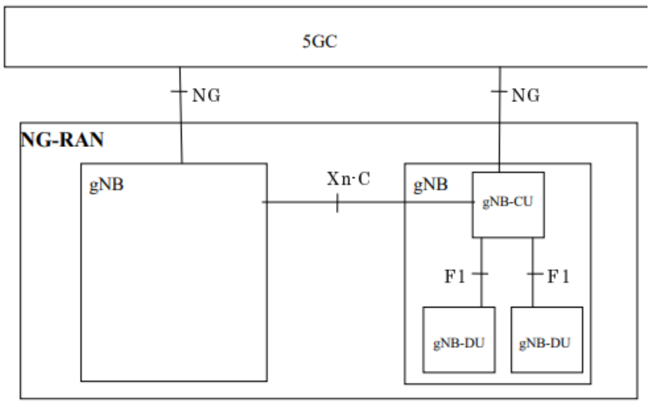


Figure A.5: Overall Architecture of NG-RAN. Source 3GPP TS 38.401, [32]

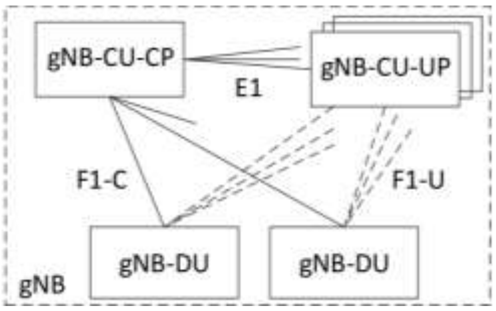


Figure A.6: Overall architecture for separation of gNB-CU-CP and gNB-CU-UP. Source 3GPP TS 38.401, [32]

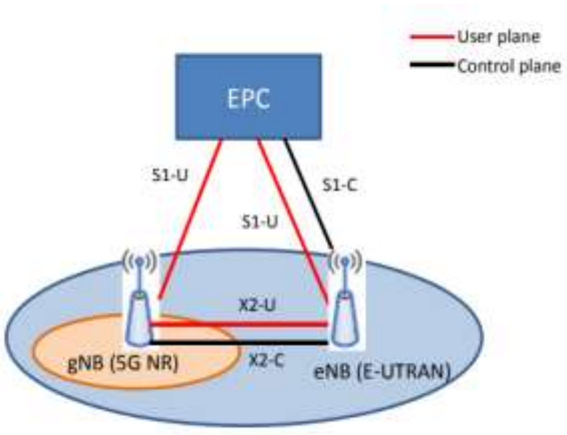
The gNB-CU is a logical node that hosts the upper layers of the radio interface protocol stack while the gNB-DU is a logical node that hosts the lower layers. Where the Interface F1 supports this functional split. Some standard bodies proposed to allocate the splicing point in the radio processing chain.

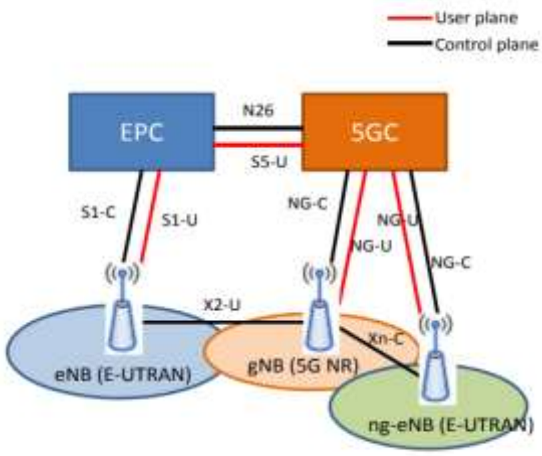
In Release 15, the split between gNB-CU and gNB-DU is between the PDCP and RLC layers (split option 2). Others approach propose the splitting between MAC-PHY (option 6) and Intra-PHY (option 7), [29].

In Release 15, The gNB-CU can be split into two entities, splitting the Control Plane (CP) and User Plane (UP) functionalities; the gNB-CU-CP logical node that hosts the RRC layer and the control plane of PDCP and the gNB-CU-UP logical node that hosts the new SDAP layer and the user plane of PDCP. Both entities are interconnected through the E1 interface, Figure A.6.

The 5G network supports multiservice capabilities this means that non-standalone and standalone deployments can co-exist, Figure A.7:

- Non-Standalone operation of 5G NR: The LTE eNB and the 5G NR nodes are interconnected to the EPC core network of LTE with dual connectivity, option 3 (section 2.1). Where LTE is used for control-functionality (e.g. initial access, paging and mobility), while 5G NR only provides user plane connectivity.
- Standalone operation of 5G NR: The 5G NG core is deployed to connect directly to the 5GC with dual connectivity with other gNBs, eNBs and ng-eNBs is possible. Where the gNB handles user and control plane functions.





(a) Non - Standalone operation (b) Standalone operation

Figure A.7: 5G deployments. Source 5G course UPC.

5G New Radio (5G NR) Interface Protocol

The 5G New Radio refers to the new radio access technologies standardized by 3GPP for 5G networks. The stack is nearly similar as the LTE layer-2 protocol stack, except for the new service data adaptation protocol (SDAP) layer of the user plane. The Figure A.8 and Figure A.9 shows the stacks protocols for the user and control planes.

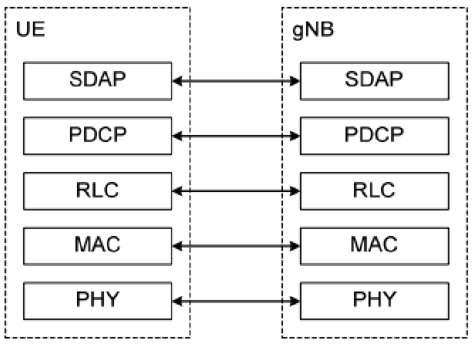


Figure A.8: User plane. Source TS 3GPP 138.300, [33]

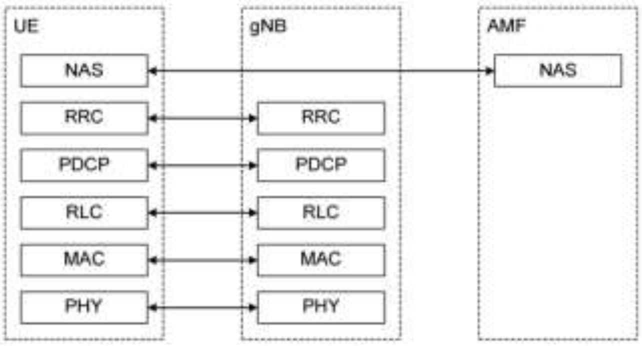


Figure A.9: Control plane. Source TS 3GPP 138.300, [33]

The layer 2 of the 5G NR have been designed to support lower delay and higher data rates in NG-RAN regardless of the connecting CN (core network). The following subsections describes the new sub-layer and the functionalities of the PDCP, RLC, MAC and PHY sub-layers that are upgraded for 5G NR.

1. *SDAP (Service Data Adaptation Protocol)*

The SDAP is introduced to performed mapping of the IP flows and radio bearers. The SDAP encapsulates the IP packets, and the header marked with an identifier indicating the QoS for those packets in both UL and DL. SDAP is not enabled in non-standalone mode.

2. *PDCP (Packet Data Convergence Protocol)*

The PDCP sub-layer performs IP header compression and decompression through RoHC (Robust Header Compression), Ciphering, removes duplicate packets and can also perform re-ordering/in-sequence delivery (useful in case of intra-gNB handover).

In LTE, in normal operation, PDCP always delivers packets in-sequence. When a packet is missing, PDCP does not deliver to higher layers any received packets which have a SN larger than the SN of the missing PDU. In principle, any packets received after the missing PDU could be delivered to higher layers (i.e. out-of-sequence delivery) if higher layers can handle out-of-sequence reception. One of the reasons why PDCP has traditionally deliver packets in-sequence has been due to TCP. TCP has been quite sensitive to the out-of-sequence reception of packets. Newer versions of the TCP protocol, however, may handle out-of-sequence reception of packets in a better way, [31].

In 5G NR, complete PDCP PDUs can be delivered out-of-order from RLC to PDCP. RLC delivers PDCP PDUs to PDCP after the PDU is reassembled. PDCP reordering is always enabled if in sequence delivery to layers above PDCP is needed (i.e. even in non-DC (dual connectivity) case). RLC receive entity needs to keep track of each packet in the window to determine if any of them has been completed and delivered to PDCP,[31].

3. *RLC (Radio Link Control)*

This sub-layer provides segmentation of packets (RLC SDU), in order to match the transmitted PDU size to the available radio resources, and error correction through ARQ.

It works in 3 modes:

- Transparent Mode (TM): Maps the RLC SDUs (i.e., PDCP PDUs) into RLC PDUs
- Unacknowledged Mode (UM): Segmentation/Concatenation of the RLC SDU at the transmitter and reassembling/reordering at the receiver and packet loss detection (since the MAC layer supports retransmission mechanism (HARQ)).
- Acknowledged Mode (AM): Same functionalities of UM plus retransmission mechanism.

Some functionalities are no longer performed at the RLC such as:

- No Reordering in the RLC layer as the reordering is done in PDCP.
- No concatenation, the RLC PDU generation processing is done before scheduling by not supporting the RLC concatenation function that multiplexes the data in the same bearer based on the TB size. If there is no concatenation, we can pre-process a PDCP PDU in to MAC SDU before the scheduling, otherwise after scheduling decision is communicated to RLC only RLC SDU will be converted to RLC PDU and given to MAC, this process introduce delay, [31].

4. *MAC (medium access control)*

The MAC layer provides multiplexing and demultiplexing of the logical channels to the transport blocks that are carried by the physical layer, priority handling between data from different radio bearers, and error correction through Hybrid ARQ (retransmitting only the portions of a packet). A notable addition compared to LTE is that the MAC protocol carries inband control signalling used for beam management, Buffer status report, timing advance, etc, within the physical layer.

5. *PHY (physical layer)*

The PHY layer supports mapping of the signal to the time/frequency resources, physical layer hybrid-ARQ processing combining different redundancy versions of the retransmitted Code Block Groups (CBG).

6. *RRC (radio resource control)*

The RRC supports control plane functionalities involving the mobile and the gNB. It includes connection establishment and release functions; the broadcast of system information; radio bearer establishment, reconfiguration and release; RRC connection mobility procedures; paging; and power control. The NR RRC support of an “on demand” system information mechanism that enables the UE to request specific system information instead of consuming radio resources provided by frequent periodic system information broadcast.

7. *NAS (non-access stratum)*

The NAS protocols terminate the UE and the AMF of the 5G core network and are used for core network related functions such as registration, authentication, paging and session management. The NAS relies on the RRC control plane for transferring the messages between UE and AMF through the gNB.

8. *RRC states*

The NR RRC introduces a 3-state model with the addition of the RRC INACTIVE state. The RRC Inactive state provides battery efficiency like the RRC Idle state but with a UE context remaining stored within the NG-RAN so that transitions to/from RRC Connected are faster and reduce the signalling overhead.

Glossary

3GPP	3rd Generation Partnership Project
5G	fifth-generation
ACK	acknowledged
ACM	Adaptive Coding Modulation
AIMD	additive increase multiplicative decrease
AM	Acknowledged Mode
AQM	Active Queue Management
BDP	bandwidth delay product
BinF	Bytes in-flight
BLER	block error rate
BSR	Buffer Status Report
CAPEX	capital expenditure
CC	congestion control
CDF	cumulative distribution function
CRRM	Common Radio Resource Management
CRT	Complete Report Table
cwnd	congestion window
DC	Dual connectivity
DL	Downlink
DTS	Downlink Traffic Scheduling
eMBB	enhanced mobile broadband
EPC	Evolved packet core
FIFO	first in first out
FS	Fast Switching
FTP	file transfer protocol
gNB	next generation NodeB
HO	Handover
HOF	Handover failure
HTTP	hyper-text transfer protocol
ITU-R	International Telecommunication Union - Radiocommunication Sector
LB	Load balancing
LOS	line-of-sight
LTE	Long-Term Evolution
MAC	medium access control
MCG	MeNB Cell Group
MeNB	macro eNB
mMTC	massive machine-type communications
mmWave	Millimetre wave
MTU	maximum transmission unit
NFV	Network Function Virtualization
NLOS	non line-of-sight
NR	New Radio
ns-3	network simulator 3
NYU	New York University
Pcell	Primary cell

PDCP	Packet Data Convergence Protocol
PDR	packer delivery ratio
PDR	Packet Delivery Ratio
PF	Proportional Fair
P-GW	PDN gateway
PHY	physical layer
PRB	physical resource blocks
PSCell	Primary SCell
RAT	Radio Access Technology
RLC	Radio link control
RLF	radio link failure
RT	Report Table
RTO	Retransmission timeout
RTT	Round trip time
RWND	received window
SB	Splitting bearer
SCG	SeNB Cell Group
SCH	Secondary cell handover
SDMA	Spatial Division Multiple Access
SEM	standard error of the mean
SeNB	secondary eNB
S-GW	Serving Gateway
SI	Study Item
SINR	signal-to-interference-plus-noise ratio
SMSS	sender maximum segment size
SRS	Sound Reference Signals
ssThresh	slow start threshold
TB	transport block
TCP	transport control protocol
TDD	time division multiplexing
TTT	time-to-trigger
UCI	Uplink Control Information
UE	user equipment
UL	Uplink
UMi	urban microcell
URLLC	ultra-reliable and low-latency communications
V2X	vehicle to everything