Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

A comparison of TCP congestion control algorithms for video streaming over LTE in autonomous vehicle scenarios

Patrick Michael Hughes
[redacted]

Project Supervisor: Dr. Meriel Huggard

May 8, 2020

A Final Year Project submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer Engineering)

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed:Patrick Hughes Date:08/05/2020

Abstract

Many TCP congestion control algorithms (CCAs) that were originally built for wired networks are still used to dictate packet transmission for flows that operate over wireless, lossy LTE channels today. These algorithms perform poorly in these channels, over reacting to loss that occurs from factors other than packet loss. This study performs a comparison of a variety of TCP algorithms in order to find one that avoids these shortcomings and utilises available LTE bandwidth. Studies such as [29] fail to realistically model the environment in which real LTE flows operate over. These studies evaluate TCP performance over LTE assuming user equipment is stationary, handover is not occurring etc. This study will attempt to emulate a real world LTE scenario that takes these factors into account. The results show that most TCP CCAs are incapable of fully utilising available LTE resources. The exception to this is TCP BIC which aggressively recovers from loss events to achieve high goodput. An honorable mention is given to TCP BBR which demonstrates the most potential of the TCP variants up for comparison.

Acknowledgements

First of all I would like to thank Patrick and Therese Hughes for providing an education that so few in this world are fortunate enough to get. Providing an education is one thing but being great parents and mentors for life is another and they need to know how well they excelled in both areas.

I would also like to thank my supervisor Dr. Meriel Huggard for being a constant source of great advice. She always focused on the positives of the work I brought to her, choosing to build on it rather than point out what I hadn't yet achieved. I wish her and her dogs all the best.

Contents

List of Figures

List of Tables

Nomenclature

ns-3	Network Simulator Three, a discrete-event network simulator for Internet systems
LTE	Long-Term Evolution, a standard for wireless broadband communication for mobile devices
UE	User equipment, any device used directly by an end-user to communicate
eNB	evolved NodeB, manage radio resource and mobility in the cell and sector to optimize a UE's communication
SUMO	Simulation of Urban MObility, SUMO is an open source, highly portable, microscopic and continuous traffic simulation package
CCA	Congestion Control Algorithm
ACK	Ackknowledgment to signal that data has been received successfully
CWND	Congestion Window, TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK, Units: B, bytes
RTT	Round-trip time is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgement of that signal to be received, Units: ms, milliseconds
Mbps	All instances stand for Goodput, the number of useful information bits delivered by the network to a certain destination per unit of time, Units: Mbps, mega bits per second.

1 Introduction

1.1 Inspiration

Transmission Control Protocol (TCP) is the most dominant transport protocol used by applications on the internet [1]. There are many different versions of TCP each with their own algorithm that determines transmission rate. Despite many of these algorithms being around for a long time and LTE becoming the dominant mode of cellular communication having been introduced in 2009, there is no definitive optimal TCP algorithm for communication over LTE.

In recent years internet traffic has been consistently growing and shows no sign of stopping. The proportions of this growth are also changing with video streaming rapidly becoming the dominant form of global IP traffic. Cisco's Visual Networking index [2] predicts that by 2022 Internet Video will be responsible for 82 percent of global IP traffic. Figure 1.1 demonstrates both the overall global IP traffic growth and the change in proportions of this growth leaning towards video.

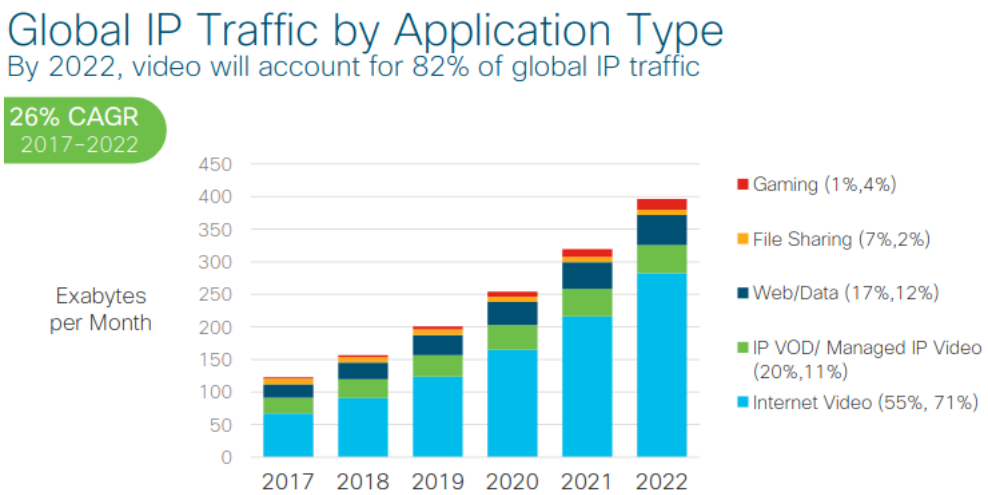

Figure 1.1: Cisco's IP traffic growth forecast for 2022. Taken from [2]

[3] predicts that by 2045 half of all new vehicles will be autonomous. Autonomous vehicles will undeniably become the norm moving into the future. The question must be raised, what will people do with their spare time on long car journeys when they no longer need to drive their cars? One likely answer to this question is stream video whether this be via Netflix, Youtube, Disney plus, Amazon Prime, Hulu etc. We already known this to be true as Tesla, one of the largest autonomous vehicle manufacturers, has already introduced "Tesla Theater" mode to the 18 inch screens that are positioned at the centred of Model S dashboards. "Tesla Theater" mode allows streaming of Netflix, YouTube, and Hulu or Hulu + Live TV. Video-streaming is a notoriously bandwidth intensive application. A standard 1080p Netflix stream requires 4300-5800 kbps with 4K requiring 8000-16000 kbps. Netflix uses TCP to stream its video.

The above factors of increased video streaming over LTE to autonomous vehicles at high speeds will lead to poor Quality of Service for these video streams. This is compounded by considering that LTE resources tend to be extremely limited on motorways, especially rural motorways. This dissertation presents an attempt to find the TCP algorithm that performs optimally in this environment and can manage to fully utilise LTE lossy channels and hence provide enough bandwidth to stream video.

1.2 Dissertation Structure

Chapter 2 will provide a literature review on related papers that were read when preparing this dissertation. It will also discuss the choosing of software components to conduct the simulations performed in this study. Chapter 3 will introduce the modules used in these simulation as well as how the simulations were setup and parameterised. Chapter 4 will present brief descriptions of the simulations performed in this study followed by their output in graphical form and an overview of the observations made on these results. Chapter 5 will conclude this dissertation followed by a brief section on future work should this dissertation be built on in the future.

2 Background

2.1 Transport Protocols

The purpose of a transport layer protocol is to provide logical communication between applications running on different hosts. On the sender side known as the server, transport protocols break application transmissions into segments and pass these segments to the network layer. On the receiver or client side, they reassemble these segments and pass the transmission to the application. In the case of the internet there are two main transport protocols, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). For the purpose of this thesis we will focus solely on TCP performance.

2.1.1 Overview of TCP

In this section a general overview of TCP is provided, with a focus on the key workings of this transport layer protocol needed to appreciate and understand the work detailed in this thesis. TCP stands for Transmission Control Protocol. It has characteristics of being reliable and delivering segments in the order they were sent. It achieves these characteristics by:

- Establishing a connection between the sender and receiver prior to transmission
- Congestion control
- Flow control

TCP is a point-to-point transport protocol meaning a connection has one sender and one receiver. This connection is established through a TCP "handshake" which involves transmission of control messages that establish initial sender and receiver states. TCP connections are fully duplexed. This means that both parties can communicate with one another in both directions by sending packets. The size of these packets is determined by the MSS (Maximum Segment Size). These packets are generated from an in-order byte stream and are also sent in-order. TCP is flow controlled which means that the sender will not overwhelm the receiver. We will discuss the mechanisms used to achieve this below.

Pipelining

TCP's connections are pipelined, this means that the sender is allowed to maintain multiple in-flight packets at a time that have not yet been acknowledged or ACKed by the receiver. The purpose of pipe-lining is to increase channel utilisation. By keeping multiple segments in flight at the same time we decrease downtime waiting for ACKs. In the following figure pipelining triples the channel utilization between client and sender.

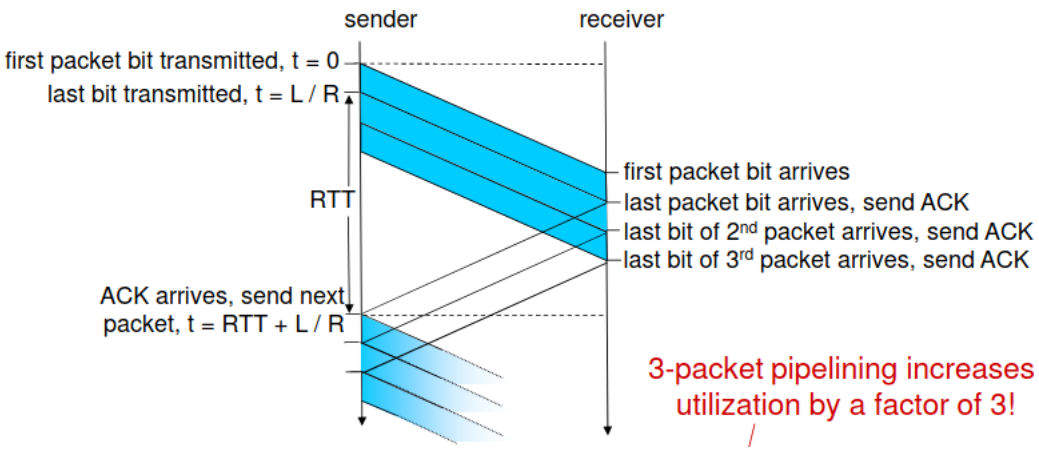

Figure 2.1: Figure illustrating how packets are sent using TCP with regards to ACKs, RTTs and responses. Figure taken from[4]

Round Trip Time (RTT) and timeout

Figure 2.1 gives an example of an RTT. Taking the y-axis on the sender side as time we can see that the RTT represents the time it took for the first packet to be sent, propagate through the connecting network, be received by receiver, be acknowledged by the receiver, for the acknowledgement to propagate back through network and finally for the acknowledgement to be received by the sender. In the work obtained in this dissertation RTT measures the delay, in milliseconds, for transmission between the remote host and UE (User Equipment).

TCP uses measurements of RTT to set the timeout value on sent packets. The timeout value is how long TCP will wait for an ACK before assuming transmission of a packet has failed. TCP uses an exponentially weighted moving average of past RTTs generated from successful packet transmissions plus a safety overhead to set its timeout interval. The moving average allows TCP to place an emphasis on more recently received RTTs in order to predict future RTTs.

Sequence number

Every packet sent via TCP contains a sequence number unique to that connection. Sequence numbers enable in-order processing of received packets on the receiver end. Sequence numbers also allow re-transmissions to occur in the correct order by the sender should packets not be received for whatever reason e.g. lossy channel. Sequence numbers are contained in the TCP packet header.

TCP Flow Control

TCP flow control allows the receiver of a TCP connection to control the sender's transmission rate so as not to overflow the receiver's buffer. The receiver achieves this by including the RWND value in the TCP header of packets sent from receiver to sender. In this way the receiver lets the sender know that it must limit the number of un-ACKed in-flight bytes/packets to the value of RWND. This allows the receiver to ensure its receive buffer will not overflow.

TCP Congestion Window (CWND)

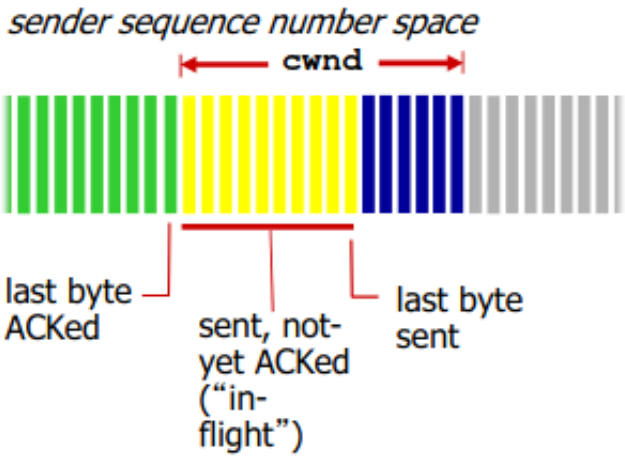

Figure 2.2: Figure illustrating operation of TCP's CWND. Figure taken from [4]

Figure 2.2 gives an example of TCP's congestion window. TCP roughly sends a congestion window (CWND) amount of data (which can be measured in packets or bytes). TCP then waits an RTT for ACKs. If an ACK is received per sent packet TCP then continues to send more packets. In this way the sender limits the transmission rate of the TCP connection. It should be noted that the size of CWND is dynamic and constantly changing according to the sender's perception of the congestion in the network the TCP connection is operating over.

TCP's transmission rate is roughly equal to:

$$rate \approx \frac{CWND}{RTT} bytes/s \tag{1}$$

The size of the CWND constantly changes throughout the lifespan of a TCP flow. The size is determined by TCP congestion control which does its best to evaluate the available performance of the current network.

2.1.2 TCP Congestion Control

What is Congestion?

We must first consider what is meant by the term "congestion" when it comes to networking. Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. In simpler terms it is when network data sources are sending too much data at once for the given network to handle.

Congestion causes queues of packets to build up in the network routers as they are not capable of forwarding packets at the rate that sources in the network are transmitting. These queues result in queuing delays as the routers take time to process the build up of packets. If the queues exceed their capacity for packets it can result in packets being dropped. Dropped packets require re-transmission from the TCP source. Delays and dropped packets can result in throughput that is less than the bottleneck link. Congestion is to be avoided at all costs because it results in in-efficient use of channel bandwidth and hence a reduction in service for all user operating over this channel.

How is Congestion handled in TCP

TCP attempts to avoid congestion using a mechanism known as end-to-end congestion control. TCP receives no explicit feedback from the network it is operating over. Instead it interprets congestion through its measures of packet loss and delay. TCP uses an algorithm that increases the sender transmission rate in an attempt to probe available network bandwidth and decreases the transmission rate in response to loss. The most basic of these algorithms is known as additive increase, multiplicative decrease or AIMD. This algorithm increase the MSS of the CWND by 1 for every successful RTT until loss occurs. When loss occurs TCP assumes it is as a result of congestion and it then performs a multiplicative decrease of the transmission rate by cutting the size of the CWND in half.

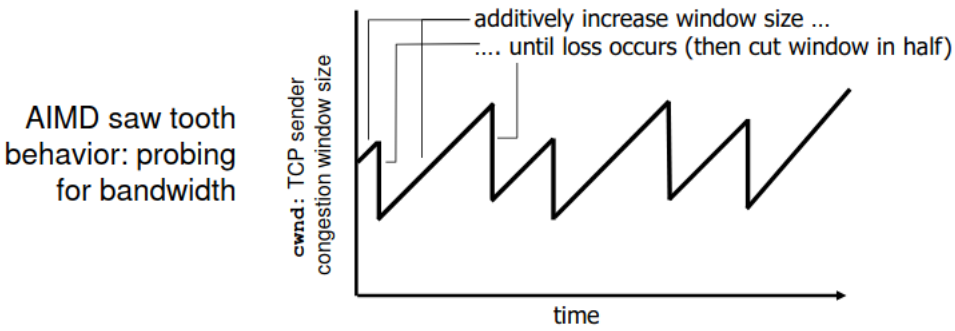

Figure 2.3: Classic saw tooth pattern of TCP CWND over time. Figure taken from [4]

Many different TCP variants have been proposed in this dissertation, two of the most widely considered are TCP slow start and TCP Tahoe. These are discussed below.

TCP Slow Start

When a TCP connection is initiated the CWND is increased exponentially until the first packet loss occurs. The CWND begins at 1 MSS and is then doubled every time an ACK is received. This results in TCP's initial sending rate being very slow but due to the exponential nature of slow start the sending rate speeds up quickly.

TCP Tahoe - An example of a basic TCP Congestion Control Algorithm (CCA)

TCP slow start resumes exponential ramp up until loss occurs. Loss can be indicated by a timeout or three duplicate ACKs. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. So when loss occurs due to timeout or 3 duplicate ACKs the CWND is set to 1 MSS in the case of **TCP Tahoe**. When loss occurs in TCP an important variable known as **ssthresh** is set to half the value of CWND before loss occurred. TCP Tahoe uses **ssthresh** to determine when to transition from exponential increase of CWND to linear increase. This transition from exponential increase to linear is also known as transitioning from the slow start stage to the congestion avoidance stage. TCP Tahoe was one of the first TCP congestion control algorithms. The above description of TCP Tahoe is one example of how a TCP CCA can go about increasing/decreasing its CWND according to perceived available link capacity or loss respectively.

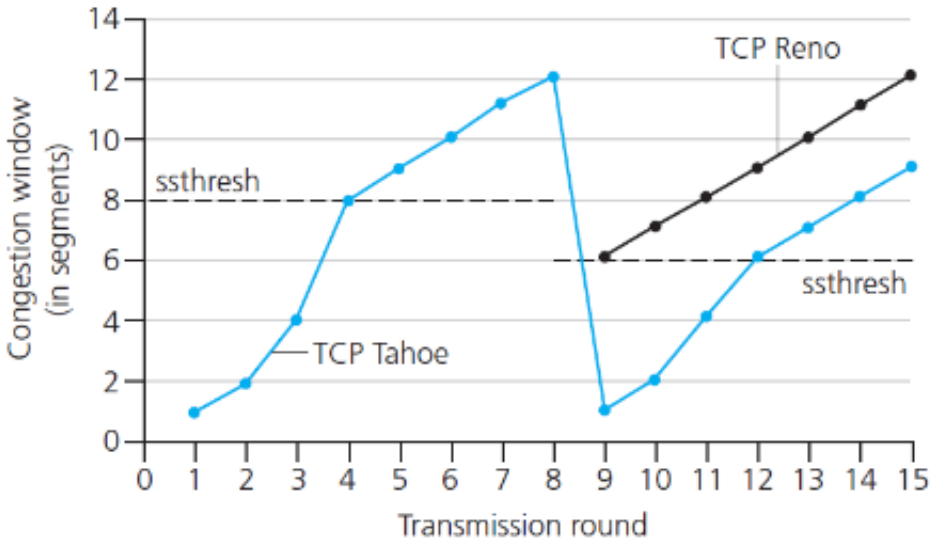

Figure 2.4: Comparison of TCP Tahoe and TCP Reno CWND changes over time. Figure taken from [4]

The main factor that differentiates TCP variants is how they react to loss and as a result calculate their value for CWND in response to this loss.

2.2 TCP variants

This dissertation focuses on an evaluation of TCP variants over LTE with regards to their reaction to loss/congestion and calculation of the CWND. This section provides an overview of the specific variants considered and the rationale for their inclusion in this work. Two specific class of TCP variants are considered; namely loss based and delay/hybrid based algorithms.

2.2.1 Loss based algorithms

TCP NewReno

TCP NewReno was first introduced to the world in 1999 by Floyd, Henderson at al.[7]. NewReno will act as a baseline for comparison in this project as it has been the most frequently implemented algorithm for years since its debut on the 4.3BSD-Reno Unix operating system. TCP NewReno uses the Additive Increase Multiplicative Decrease (AIMD) scheme that we discussed earlier in section 2.1.2. When TCP NewReno receives an ACK, it increases its CWND by a constant of 1 MSS in slow start and 1/CWND in congestion avoidance. When a congestion event occurs, which is signaled by either the receipt of three duplicate ACKs or a timeout, the CWND is decreased by a multiplicative factor and is cut in half or set to 1 segment in the case of a Retransmission Time Out (RTO).

Perhaps the most important feature of NewReno to note is that it is RTT-synchronised. This means that CWND can only increase when an ACK is received. This leads to poor results in scenarios where large values of RTT are encountered. An example of a scenario where this might occur would be when communicating over a Long Fat Network or LFN which are fat in terms of available bandwidth and long in terms of distance, and, hence delay.

Say for example we have a 10Gbps link with a 30ms RTT. It would take TCP NewReno approximately 9.7 hours to get the TCP send rate up from the initial 5Gbps sending rate to 10Gbps assuming the TCP sending rate was increased by the default maximum segment size of 536 bytes per RTT. This also assumes that no packets are dropped over this entire duration which is incredibly unlikely.

TCP Westwood

TCP Westwood was introudced in 2001 by Mascolo et al.[8]. TCP Westwood is actually a sender-side only version of TCP NewReno. It employs the Additive Increase Adaptive Decrease scheme (AIAD) for congestion control. This means that when congestion occurs, TCP Westwood tries to estimate the network's available bandwidth and use the estimated value to adjust the CWND instead of performing a flat halving of the CWND as is the case for TCP NewReno. TCP Westwood attempts to estimate the available network bandwidth when it receives an ACK be using the ACK stream for information on what to set the CWND and ssthresh values to. Similar variants such as Westwood+ attempt to estimate available bandwidth every RTT.

TCP BIC

TCP BIC [9] presented in 2004 by Xu et al., otherwise known as Binary Increase Congestion Control is a TCP version that uses a binary chop search algorithm to probe for a packet sending rate that sits on the threshold of triggering packet loss. When loss occurs BIC remembers the previous maximum value for CWND in the current flow, as well as the current CWND value.

Every time an RTT interval occurs without loss, BIC attempts to inflate its CWND value by one half of the difference between the current CWND and the maximum recorded CWND for the current flow. This allows BIC to quickly recover from the previous loss event. In addition as BIC approaches the old maximum value for CWND it slows down the rate at which it increase CWND. It achieves this by halving the rate at which it increase CWND upon each RTT.

BIC also limits the rate at which it increase CWND through the use of a maximum inflation constant to limit the maximum rate change that can occur per single RTT interval. Overall this results in BIC behaving like a hybrid between a linear and a non-linear response. The initial CWND increase after loss occurs is a steep linear increase, however as CWND approaches the point where packet loss last occurred, the rate at which CWND increases slows down.

TCP CUBIC

TCP CUBIC is a refined version of TCP BIC. TCP CUBIC was presented in 2008 by Xu et al.[10]. The same authors who created TCP BIC. TCP CUBIC uses a third-order polynomial function to determine the rate at which CWND increases. This is in contrast to the exponential function used by BIC. The cubic function itself is a function of the time since a loss event last occured. This cubic time function is used to determine when TCP CUBIC will increase CWND as opposed to TCP BIC implicitly increasing CWND upon RTTs. This use of a time interval instead of an RTT counter allows TCP CUBIC to behave fairer in scenarios where multiple flows are operating with different RTTs.

2.2.2 Delay/Hybrid based algorithms

The remainder of the algorithms used for the comparison study presented in this dissertation are delay/hybrid based. This means that rather than using a loss event as the signal for congestion, they use an increase in delay or RTT measurements to determine when congestion is occurring and, consequently, reduce their sending rate. It was important to include such algorithms in this study as LTE channels are lossy channels and so it could be hypothesised that loss based algorithms would perform poorly. In particular it was considered likely that loss based algorithms would over-react to loss as a result of propagation/fading of the signal rather than from loss due to packet drops as a result of congestion.

TCP Vegas

TCP Vegas was first proposed in 1994 by LS Brakmo, et al. [12]. It was new in the sense that it used delay in the form of RTT to determine when congestion was occurring in a TCP flow. In its simplest form TCP Vegas reduces its CWND when values of RTT are increasing. When values of RTT remain steady over time TCP Vegas increases its CWND to probe the network and find the point were RTT values are beginning to increase again.

Because TCP Vegas' sending rate is adjusted in response to received RTTs its adaptation is essentially linear over time. This linear adaptation rate makes for very slow adaptations to large bandwidth resources being made available.

TCP Vegas also has more serious problems in the form of being out competed for resources by other algorithms operating in the same channel. TCP Vegas is commonly out competed for resources because it adapts its CWND values earlier than loss based approaches. For example a loss based algorithm like NewReno will not adjust its CWND until it detects packet loss from queues that have overflown. Whereas TCP Vegas will lower its CWND at the onset of queuing. The concurrent loss based algorithm will then adjust to use these resources left behind by TCP Vegas's adaptation. This will further signal TCP Vegas to reduce its sending rate and so on.

TCP Yeah

TCP Yeah [13] has two modes of operation: "Fast" mode and "Slow" mode. During the FAST mode TCP Yeah aggressively increases the CWND according to the Scalable TCP (STCP) rule introduced by Kelly, [14]. STCP performs the following operations on CWND:

$$CWND := CWND + 0.01$$

for each ACK received while not in loss recovery and

$$CWND := 0.875 \times CWND$$

upon a loss event. This allows TCP yeah to recover from loss over a 10Gbps link with a MSS of 1500 bytes and a round trip time of 200ms in approximately 2.7 second when it would take TCP NewReno 4hrs and 45mins. During the slow mode TCP Yeah is essentially just TCP Reno. The mode that TCP Yeah is in is decided by estimating the number of packets in the bottleneck link queue.

TCP BBR

TCP BBR [15] is a delay controlled TCP CCA developed by Google and released in 2016. BBR aims to operate at the point in which the bottleneck in the path it is operating over is about to begin queuing. BBR is constantly calculating continuous estimates of the current TCP flow's RTT and bottleneck send rate. By bottleneck send rate we mean maximum rate to send before queuing at the bottleneck link of the path occurs. The esimation for RTT of the link is taken as the minimum of the last 10 RTTs recorded by BBR. The bottleneck send rate is estimated as the maximum rate of delivered data by comparing the sent data stream to the ACK stream.

BBR implements pacing of its packets that corresponds to the perceived bottleneck rate. This is intended to prevent network queuing at the bottleneck link. Traditionally the network would perform rate adaptation at the bottleneck link as other algorithms did not take this into account when increasing their CWND. For example NewReno tends to send packets in a burst at the RTT interval which relies on the network to rate adapt this data stream to the available speed at the bottleneck link which causes queuing at the bottleneck link.

2.3 Initial Research

2.3.1 Learning about BBR

The initial goal of this project was to "use ns-3 [5], a powerful and popular simulator used for network research to explore the impact of BBR on end-user QoE" in video transmission. BBR is a TCP congestion control algorithm developed by Google in 2016. BBR has been found to produce higher throughputs and lower delays [15] in certain scenarios when compared to traditional common TCP variants like TCP CUBIC and TCP NewReno. Delay-BBR was being proposed specifically for video transmission. The initial goal of my project was to evaluate the quality of experience of video transmitted by Delay-BBR in ns-3.

Thus I began my research by reading [19]. This describes an implementation of the BBR algorithm in ns-3 and validating that it behaves and performs similarly to real BBR implementations such as those found on the linux kernel or those implemented by Google.

Why BBR is different and how it works

The goal of BBR is to operate a TCP flow at the point of onset of queuing at the path bottleneck. This is in contrast to traditional algorithms which operate at the point of loss occurring as a result of bottleneck queue overflow.

BBR achieves its goal when the number of bytes-in-flight is equal to what it calls the bandwidth delay product or BDP. BDP is calculated by:

$$BDP = B_{\text{max}} \times R_{\text{min}} \tag{2}$$

Where R_{min} is the minimum RTT estimate received in the past 10 seconds and B_{max} is the largest bandwidth estimate in the past 10 RTTs. Every time BBR receives a packet it estimates round-trip and bandwidth for this packet and adds them to their respective windows for continual R_{min} and B_{max} updates. In this way BBR is constantly modelling the capacity of the link it is operating over.

BBR has four possible states it can be in at any one time. These states determine, for example, pacing rate, probing for available link bandwidth and CWND.

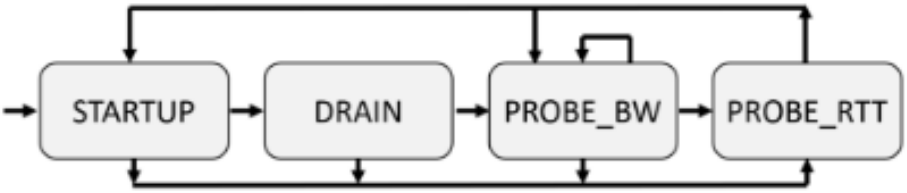

Figure 2.5: BBR state transition diagram. Image taken from [19]

These states are STARTUP, DRAIN, PROBE-BW and PROBE-RTT and they work as such:

- **STARTUP**: A BBR flow starts in STARTUP phase. In this phase BBR ramps up its sending rate by the congestion window to $BDP \times \frac{2}{\ln 2}$, roughly doubling the bit rate per round-trip time.
- **DRAIN**: BBR enters DRAIN state when bandwidth has not increased by more than 25 percent over last 3 RTTs. When in DRAIN BBR reduces pacing rate but keeps CWND the same in order to DRAIN the queue that built up at the bottleneck link in STARTUP phase. It then enters PROBE BW.
- **PROBE BW**: BBR flows remain mostly in this state. Sending at the bottleneck link rate. PROBE BW cycles once per round-trip time, through a series of 8 gain values:[1.25,0.75,1,1,1,1,1,1]. These values are applied as multipliers to the bottleneck rate in a cycle. 1.25 attempts to probe for available bandwidth.
- **PROBE RTT**: If BBR does not received an RTT sample that decreases the R_{min} for 10 seconds, then it enters the PROBE RTT. BBR reduces in flight packets by 98 percent in order to re-probe the link's delay. BBR stops probing after one round-trip time or 200 milliseconds, whichever is longer.
- Note: When BBR exits PROBE RTT state it enters PROBE BW if the full band-width estimate of the pipe has been reached. Otherwise, it enters STARTUP to try to re-fill the link.

BBR uses the above methods to react to congestion as opposed to traditional TCP CCAs which react to loss. This allows BBR to ignore loss as a sign of congestion, choosing instead to operate according to its BDP model. I hypothesised that this approach would have major benefits over traditional TCP CCAs in scenarios where loss occurs for reasons other than buffer overflow such as lossy LTE channels.

Choosing a version of BBR for my project

It should be noted that the implementation of BBR detailed by Claypool et al [19] is the implementation used when integrating BBR into the simulation scenarios considered in this dissertation. BBR is not native to ns-3 and so it needs to be imported into ns-3 for use in simulations. The first BBR implementation considered in this study was [16]. However, during simulations conducted for this study it produced very low throughput for every scenario considered. After much effort to determine the cause of this poor performance, it was decided to use an implementation from Claypool et al.[19] implementation. This work was presented in 2018 at WNS3, the official ns-3 workshop held every year. Thus both implementations were considered reputable enough to be trusted for use in this study.

Due to the nature of how BBR probes for more bandwidth it is interesting to consider how it interacts with other existing congestion control algorithms should they share the same channel. This led to a shift in the focus of the research done for this dissertation.

2.3.2 Channel Fairness across Congestion Control Algorithms

The literature review then focused on how BBR interacts with other existing algorithms. [20] showed that BBR and CUBIC do not share bandwidth fairly. This was attributed to BBR overestimating its BDP parameter.

[21] divided algorithms according to their design philosophy by loss, delay and hybrid nature. It found that BBR favours a flow with a higher RTT at the expense of other flows and while doing so doesn't maintain the low queuing delay that the algorithm promises. [22] found that in most cases BBR does not share bandwidth in a fair manner with loss-based algorithms such as Reno or Cubic.

Based on this, fairness amongst CCAs was considered as a possible direction for this project. However, after reading CISCO's growth forecast for LTE video traffic [2] it was decided that the focus should be on congestion control over LTE. In this context fairness comparisons among CCAs are of less importance. This was mainly because LTE assigns resource blocks to users, so fairness is less of an issue here as it is somewhat predetermined.

2.4 LTE

2.4.1 LTE Network Summary

LTE also known as Long Term Evolution or the E-UTRAN (Evolved Universal Terrestrial Access Network) is a mobile communications standard developed by 3GPP. LTE is the access part of the Evolved Packet System. This network can be characterised as having high peak data rates, high spectral efficiency and short round trip time. It is also found to be flexible in how it allocates bandwidth to UEs [23]. This flexible nature is due to each eNodeB possessing a fixed number of Resource blocks of which it can allocate to a connected UE.

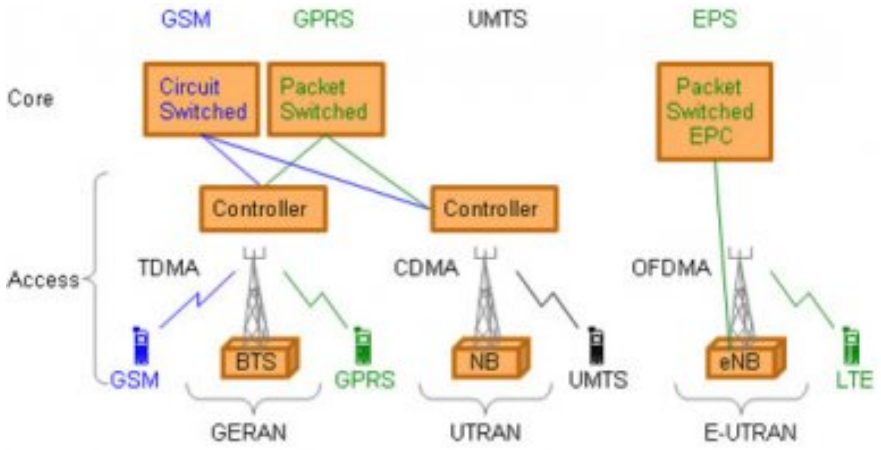

Figure 2.6: The transition of network solutions overtime from GSM to LTE. Figure taken from 3GPP [23]

The Evolved Packet System is IP based. IP addresses are allocated when a UE is turned on/is in range of an eNB and released when a UE is switched off/out of range of an eNB. The LTE access network is composed of a network of base stations. There is no centralized point that maintains control over the eNBs, instead the eNBs are inter-connected with one another using the X2-interface. Each eNB is also connected to the EPC via the S1-interface.

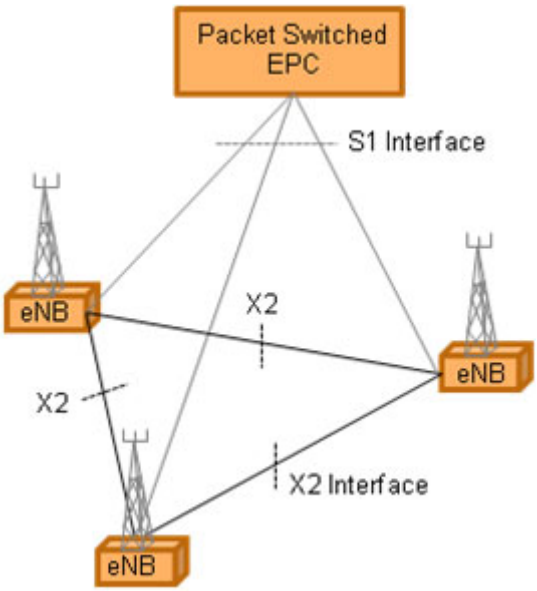

Figure 2.7: Diagram showing eNB interconnection via X-2 interface and connection to EPC via the S1 Interface. Figure taken from 3GPP [23]

Intelligence is intentionally distributed in an LTE network among the eNBs as opposed to having a single centralized point. It is done this way to make the handover process of a UE from one eNB to another smoother and faster than if they had to communicate to a centralized point in the core network. Many UE applications are delay dependant such as voice-chat, online gaming or potentially video-streaming. In the case of these applications it is pivotal that the time it takes for connection transfer from eNB to eNB is kept to a minimum.

2.4.2 The EPC Core

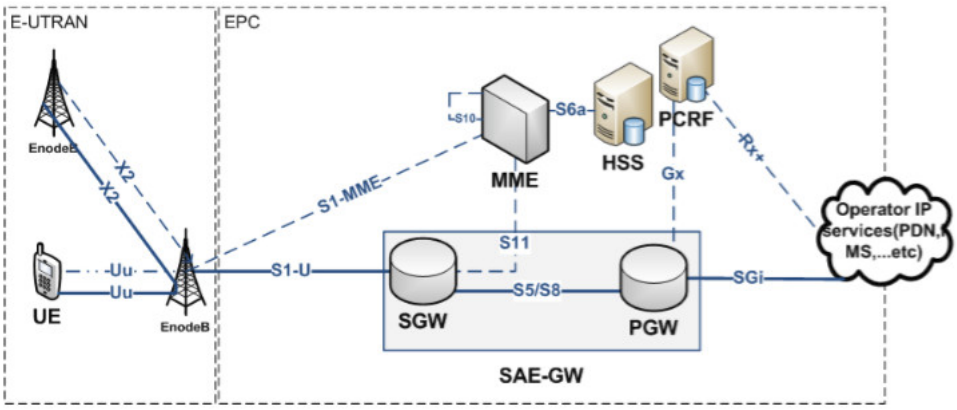

Figure 2.8: Division between E-UTRAN and the EPC core. Figure taken from [24]

The EPC represents the Core of an LTE network. It is formed by multiple nodes, the main ones being MME, SGW, PGW and HSS. These nodes offer multiple points of functionality like mobility management, authentication, session management, setting up bearers and application of different Quality of Services [25]. The EPC is composed of the following six nodes:

1. **Serving Gateway (SGW)**: The Serving Gateway connects the E-UTRAN to the EPC. It is responsible for connecting UEs to the external Internet Protocol Networks. It performs routing on incoming and outgoing IP packets. In this regard it serves as a place of reference to a UE when it is handed over from one eNB to the next.

2. **PDN Gateway (PGW)**: The PGW connects the EPC to external IP networks and routes traffic between them. It performs other tasks such as packet filtering, allocating IP addresses to new UEs and policy enforcement. (Note: In real world EPC implementations the SGW and PGW are commonly found in the same network device).

3. **Mobility Management Entity (MME)**: is the main control node in the EPC. It handles signals between UEs and network within the EPC core. It also handles signalling between eNBs and the EPC core. MME performs authentication on UEs by traversing with the HSS. It also keeps track of the mobility of UEs allowing them to access the network and keep track of their state and location.

4. **Home Subscriber Server (HSS)**: acts as a central database that holds details about a subscriber's information and UE authentication. It also performs other tasks such as call and session setup.

5. **Policy and Charging Rules Function (PCRF)**: deals with charging in the EPC according to a UE's policy e.g. making sure a UE doesn't exceed their data limit or has paid their bill this month.

The Radio Access Network or E-UTRAN controls radio connections between the eNBs and UEs. The eNBs connect UEs to the EPC via a radio interface.

For the purpose of the simulations carried out in this study the functionality of the EPC Core is implemented through the use of the LENA module [26]. This module comes natively with ns-3. For more information on its integration into ns-3 see [27]. As of ns-3 version 3.30 (August 2019) SGW, PGW, MME are now implemented in their own full nodes. These are automatically instantiated by the user when calling certain LTE setup functions in a script. This module is discussed in further detail in the Section 3 of this dissertation.

2.5 Congestion Control algorithms over LTE

BBR-enabled network coding for car to cloud data transfer [28] highlights the fragile nature of delivering vehicle data to the cloud or vice versa cloud to vehicle. This happens because vehicles move fast in their environment and channel conditions vary heavily. This fragile connection does not bode well for future autonomous vehicle passengers who are likely to stream video when they no longer need to drive. Solutions to this fragile connection became one of the main focuses of this dissertation.

It was hypothesised that there might be an optimal CCA for vehicles in these channel conditions. [29] compares how the five most common TCP variants utilize radio resources over LTE networks. This inspired some aspects of the research carried out for this dissertation. The simulations reported on in [29] were quite basic and it was felt that they could be improved upon in several ways:

1. They should have included a CCA that was specifically developed for cellular channels. The SPROUT [18] algorithm uses stochastic forecasts to achieve high throughput and low delays in cellular networks. This was considered for use in this project.

2. The simulations in [29] only involved one eNB. Handovers in LTE are very common and should be accounted for in the simulation. Vehicles move very fast through their environment and handovers between eNBs are guaranteed to happen, especially over the course of a video transmission. This project takes handovers between multiple eNBs into account.

3. The mobility model used in [29] was extremely limited. The UE they selected for analysis moved in one dimension from the eNB. The other UEs that generated background traffic to the eNB were also stationary for the duration of the simulation. It was felt this made the results unrealistic.

2.6 Network Simulator 3 - ns-3

ns-3 [5] is a discrete event network simulator and is the primary software component of this study. ns-3 is composed of a set of libraries that implement different networking functionalities such as the WiFi library or LTE library. The user writes a script that links libraries that correspond to the network scenario they are attempting to simulate. The user then runs this script in order to extract network performance data for post analysis and research. I chose ns-3 for network simulation for multiple reasons:

- It is highly regarded amongst academics and researchers in the field of Networking.
- It was the most common software used for network simulation in the papers I read that compared TCP variants over LTE.
- It has a well documented API and in-depth tutorials.
- Allows me to perform experiments that are difficult/impractical to perform with real networks.
- Capable of maintaining reproducibility of simulations, something that is difficult to maintain over real networks.
- It was recommended by my supervisor.

In ns-3 the user can write their simulation scripts in C++ or Python. I chose C++ as this is more commonly used amongst the ns-3 community and therefore I would have a broader range of support to pull from should I need help. C++ also resulted in performance benefits which would be necessary when simulating complex LTE scenarios in a reasonable time.

ns-3 presents common Networking elements in the form of four key abstractions, these are Nodes, Applications, Net Devices and Topology Helpers. A Node represents a base computing device upon which we install Net Devices. Net Devices enable a node to communicate with other Nodes in the network over Protocols via Channels. Topology Helpers are used to link Nodes via Channels and parameterize these Channels using the ns-3 Attribute system. For example a point-to-point channel can be parameterized with a DataRate, Delay, ReceiveErrorModel etc. Next Applications are installed on Nodes in order to create traffic on the network. Finally we set a start/stop time for the simulation and then run it. Almost all ns-3 simulations follow this order to some extent.

ns-3 allows for a high degree of customization via its ns-3 Attribute System. This permits the user to tailor the LTE simulation environment to be as realistic as possible. More detail on how ns-3 allows for this to be achieved is provided in section 3.3.1.

It should be noted that the necessary precautions were taken when beginning this project to confirm that the ns-3 build was correctly configured in order for the results to be comparable to those of other researchers using this software. To do this I:

- initially built ns-3 in debug mode and ran it with the –enable-tests and –enable-examples.
- made sure it past 100 percent of integrated tests other than those intended for optional ns-3 modules.

```
PASS: Example examples/wireless/mixed-wired-wireless.py
PASS: Example src/flow-monitor/examples/wifi-olsr-flowmon.py
PASS: Example src/wifi/examples/wifi-manager-example --wifiManager=Ideal --standard=802
erverShortGuardInterval=1600 --clientShortGuardInterval=1600 --serverNss=4 --clientNss=
PASS: Example src/wifi/examples/wifi-manager-example --wifiManager=Ideal --standard=802
erverShortGuardInterval=3200 --clientShortGuardInterval=3200 --serverNss=4 --clientNss=
633 of 636 tests passed (633 passed, 3 skipped, 0 failed, 0 crashed, 0 valgrind errors)
```

Figure 2.9: Screenshot of ns-3 test output showing 100 percent pass rate.

TCP CUBIC was chosen for comparison in this study however TCP CUBIC is not native to ns-3. CUBIC implementations were written by ns-3 users were researched. The following paper [11] by Levasseur et al. gave an account of a CUBIC implementation that had been written and tested to perform similar to real CUBIC implementations. The code for this implementation can be found here. This implementation was deemed reputable as it had been submitted to WNS2014 which is ns-3's yearly workshop. It should be noted that this CUBIC was written for ns-3.27 and this study used ns-3.30. However when this implementation was integrated into this studies ns-3 base it appeared to produce sensible results so it was used in the project.

2.7 Researching a realistic mobility model

ns-3 comes with multiple basic mobility models [30] that can be installed on nodes in a simulation. These include constant position mobility models, constant velocity models, accelerating models, random direction 2D models etc. The goal was for this studies simulations to be as realistic as possible. Therefore it needed to emulate real-world vehicle traffic that would navigate realistic road networks. In particular the plan was to emulate rural highways as opposed to city centres as this is where LTE resources are limited the most. It was felt it would be more likely to see a differentiation in the performance of TCP algorithms given the lesser LTE resources on rural highways.

The first mobility model considered simulating VANETs on a highway [31]. In this paper the authors describe how they implemented models such as an IDM car-following model and a MOBIL lane-changing model. They also implemented a highway class that simulates a straight highway and manages mobility of all vehicles on said highway. This paper presented its source code for use in NS-3. It was decided against using this model because a straight highway model was considered unrealistic.

Real-world highways are not perfectly straight and are usually surrounded by other meandering roads which could contain vehicles with potential to use LTE network resources.

A detailed literature was then conducted in order to find a sufficiently realistic mobility model for the simulations in this study. This literature involved reading [32], [33]. Both of these papers involved the use of a software package called SUMO [6]. Both papers used SUMO to integrate SUMO generated traffic into NS-3 via SUMO generated trace files. After reading about the level of granularity to which SUMO enabled [32], [33] to model traffic in their simulations, SUMO [6] was considered the most suitable mobility model for the simulations in this study.

2.7.1 Simulation of Urban MObility

SUMO [6] stands for Simulation of Urban MObility. SUMO is an open-source, multi-modal traffic simulator. It allows the user to simulate how a pre-applied traffic demand which is made up of individual vehicles will move through a road network. For example Figure 2.10 shows a sample SUMO simulation of Dublin City Centre that was created when learning how to use SUMO. This simulation contains cars, trucks, buses, cyclists and pedestrians. Over the course of its 1500 second duration 970 cars and 764 pedestrians traverse the network.

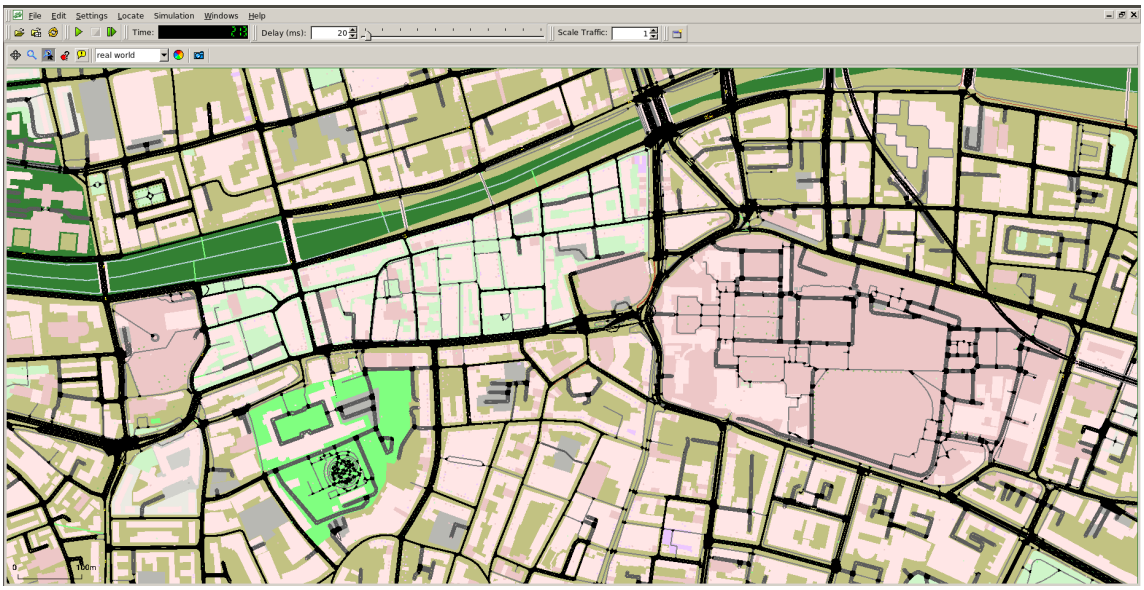

Figure 2.10: Sample SUMO simulation of Dublin City Centre

SUMO takes a microscopic approach to modelling traffic as each individual vehicle is explicitly modelled with the route that it uses to move through the network. The vehicles in SUMO are space-continuous and time-discrete meaning updates of their current position, velocity etc occur every second by default. SUMO simulations have a high level of granularity with the inclusion of vehicle types, multi-lane traffic, lane changing and importing of public transport.

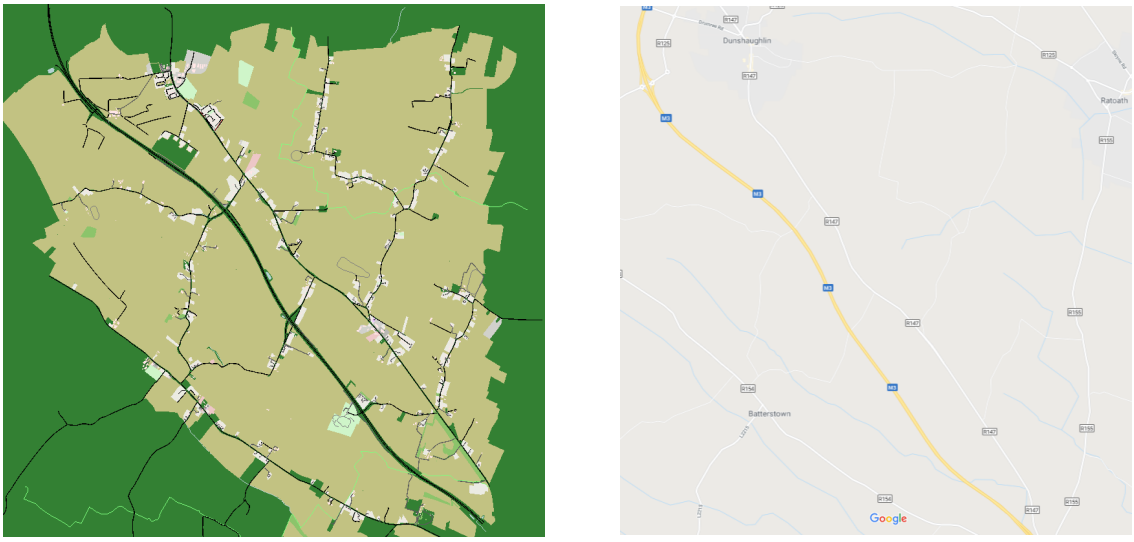

Figure 2.11: Comparison of M3 motorway segment from Dunshaughlin town to Black Bull roundabout captured in SUMO simulation to Goggle maps image of same motorway.

SUMO is capable of producing networks with several tens of thousands of edges or streets and up to 100,000 vehicles at a time.

2.7.2 How SUMO was used in this study

SUMO was used to create a traffic simulation of the road network that surrounded the segment of M3 motorway from Dunshaughlin town to Black Bull roundabout. This section was chosen as it was known from first hand experience of attempting to stream video in this area that LTE resources were limited.

In order to create this SUMO simulation a tool called osmWebWizard was used. This tool launches a web application that allows you to select a square section of OpenStreetMap and use that as the road network for your simulation. It also allows you to parameterise the simulation. These parameters include setting vehicles types in your simulation for example cars, buses, pedestrians etc, importing public transport and setting the through traffic factor of your simulation. The Through Traffic Factor defines how many times it is more likely for an edge at the boundary of the simulation area to be chosen compared to an edge entirely located inside the simulation area. A big value for the Through Traffic Factor implies that many vehicles depart and arrive at the boundary of the simulation area, which corresponds to a scenario with a lot of through traffic.

Once the desired parameters were chosen the simulation was downloaded and ran in sumo-gui. An additional vehicle was added to the simulation which entered the network at the beginning of the motorway segment and exited at the opposite end of the motorway segment. This would be the test node that network performance would be monitored from.

The simulation was then exported in xml format, converted to a tcl file and imported into the ns-3 simulations via an Ns2MobilityHelper object. This imported SUMO trace file would now function as the mobility model for all of the UE nodes in the ns-3 simulations.

3 Methodology

In order to evaluate the performance of TCP CCAs over limited LTE resources on motorways an experiment had to be designed that could accurately model this scenario. In this chapter detail will be given on the approach taken to model said scenario.

3.1 ns-3 LTE Module - LENA

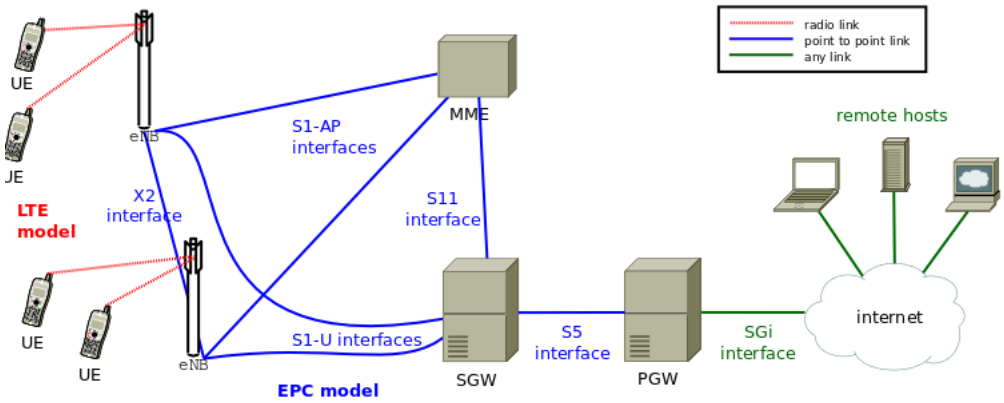

Figure 3.1: Overview of the LTE-EPC simulation model. Taken from [26]

LENA [26] is a highly regarded open source LTE/EPC Network Simulator that allows LTE small/macro cell vendors to design and test self organized Networks. The LENA network simulator has been integrated into ns-3 and makes up its LTE Module. This ns-3 LTE module allows simulation of both the E-UTRAN and the EPC core which is described in section 2.4.1.

The LTE Model (E-UTRAN) is composed of the UE and eNodeB nodes which communicate with the EPC model (EPC core). The EPC model is composed of the PGW, MME and SGW nodes which, as of ns-3.30, are all represented by their own individual nodes in the ns-3 topology. These nodes communicate over the same protocols found in real LTE Networks allowing them to serve the same functionality discussed in section 2.4.2. The LTE model is capable of instantiating tens of eNodeBs and hundreds of UEs at a time. Increasing the complexity of the simulation to this degree greatly increases the time to execute of a simulation.

Using the equipment available, it took thirty hours to execute every run of each algorithm for Scenario 2 described in section 4.2 which involved nine eNodeBs and fifty-one UEs.

LENA further emphasizes realism by implementing the resource block allocation and granularity features of LTE that were discussed in section 2.4.1. Further detail will be provided on how the eNodeB resource blocks were parameterised in the simulations later. With the inclusion of the EPC model in the simulation it is capable of modelling an end-to-end LTE IP network. For example in Figure 3.1 of the LENA architecture we have remote hosts on the right side that represent the internet which are capable of sending IP traffic to our UEs over the EPC and LTE models. By installing traffic generating applications on these remote host nodes one can simulate the transmission of IP traffic, such as video streaming, to the UE nodes over an LTE network. These applications are capable of communicating over TCP and UDP.

The LENA module is also capable of implementing LTE handover procedures between eNodeBs. Handover was important to include in my simulation as it was important to capture TCP behaviour when mobility was very high i.e. on a motorway. In this scenario handover is guaranteed to occur in real-life and so needed to be a part of my simulation. LENA gives the user the option of achieving this through manual pre-determined handovers or automatic handovers. The automatic handovers were implemented by installing the LTE X-2 interface on the eNodeBs in my simulations. As previously discussed, this interface allows eNodeBs to communicate with one another without a centralized link.

The ns-3 LENA module integration also provides added realism in the form of Propagation Models [36] and Fading Models. Propagation models attempt to simulate loss in LTE networks from phenomena such as propagation loss in free space. They achieve this by calculating the Rx signal power considering the Tx signal power and the mutual Rx and Tx antennas positions. They are installed on ns3::LteHelper objects.

LENA includes a trace-based fading model derived from the one developed during the GSoC 2010 [37]. The LENA module comes with pre-calculated fading traces that can be installed on ns3::LteHelper objects in simulations to take fading into account. LENA also provides a Matlab script that allows you to generate a fading trace file that corresponds to the parameters of the simulation.

3.2 LTE Network Topology

The LTE simulation for this dissertation was written in C++ using ns-3 and its accompanying LENA LTE module. A strong emphasis was placed on emulating realism in this LTE network. There are a total of three simulations in this study each of which has their own simulation scenario. The main simulation of this study is Scenario 3. Scenario 1 and Scenario 2 were intended to act as references for how our TCP variants would react to baseline conditions and moderate traffic under non mobile conditions.

When designing Scenario 3, the main simulation of this study, the intention was to re-create a segment of rural M3 motorway and the exact LTE resources that are dispersed around it as accurately as possible. This resource strained rural environment was modelled in order to try and answer the question of which TCP variant could best utilise the strained resources at hand. In other words which of our TCP variants could produce the highest quality video stream to an autonomous vehicle under these strained network conditions.

3.2.1 General simulation topology

The modelling of all three scenarios are quite similar and begin as follows. The ns-3 attribute system is used to set the default values of many LTE parameters prior to instantiation of ns-3 objects. Further detail on exact parameterisation values is given below. Next the value for TCP variant from the commandline is used to set the TcpL4Protocol SocketType i.e. the TCP algorithm in use during this simulation run. Remote Host nodes, eNodeb nodes and UE nodes are then declared. The EPC and LTE models are instantiated through their corresponding ns-3 helper objects. These helper object automatically instantiate the EPC PGW, SGW and MME nodes. The remote hosts are linked directly to the PGW. Internet protocol stacks are installed on remote hosts and UEs. IP addresses are assigned to nodes in simulation and IP routing between remote hosts/EPC model and UE/EPC model is performed. UEs are attached to eNodeBs. X2-Handover interface is declared. Network Traffic is setup.

3.2.2 Mobility in Scenario 3

This is the main area where simulation topology changes amongst the three simulation scenarios in the study. Detail will be given for Scenario 1 and 2 arrangements in the Results chapter. This subsection will focus on mobility in Scenario 3 as it is the main simulation in the study.

In scenario three there are a total of 9 eNodeBs and 51 UEs. The fifty first UE is the test node. This is the node that performance metrics will be measured from. All 51 UEs in the simulation have their mobility model determined by an imported SUMO trace file for the duration of the simulation.

This means their velocity and acceleration as well as their initial location are all determined by this trace file. SUMO determines vehicle speed as the minimum of many factors such as max vehicle speed, lane speed, speed in relation to vehicle ahead, intersections, stop signs, lane changing etc. The factor that is currently the minimum speed is what SUMO chooses as the current speed of a vehicle in its simulations. The trace file was exported from a SUMO simulation that accurately modelled the road network including and surrounding the M3 motorway segment from Dunshaughlin town to the Black Bull roundabout pictured in Figure 3.4.

50 of these UEs navigate this road network with a pre-determined path that was generated at random, traversing from one pre-determined edge of the network to another over the duration of the simulation. However, the test node has a user determined fixed path which enters the road network at the beginning of the M3 motorway segment in the top left corner of Figure 3.2 and finishes the simulation in the bottom left corner of Figure 3.2 at the opposite end of the M3 motorway segment.

The eNodeBs in this scenario are given fixed mobility models meaning they do not move for the duration of the simulation. Their locations were taken from the Comreg site viewer website [17] for Ireland. These latitude and longitude co-ordinates were then mapped to cartesian co-ordinates in metres of the current SUMO simulation. In this way the scale of the model is preserved and eNodeBs in cartesian co-ordinates are the same distance in metres away from one another as they are in real-life.

Figure 3.3 represents the topology of Scenario 3 in ns-3 co-ordinates. In the top left of Figure 3.3 we see the remote host and EPC core nodes (MME, PGW and SGW). These nodes are linked via point-to-point channels so delay is preset and not determined by distance hence no need for accurate spacing. The nodes that are in the center of the blue circles are eNodeBs, their cartesian co-ordinates can be seen on the Figure axes. Each eNodeB has as X2-interface link to other eNodeBs for handover purposes and a direct link to the EPC model. Figure 3.3 is a screenshot taken from a NetAnim playback of Scenario 3. NetAnim is an offline network animator that can take XML digests of ns-3 simulations and play them back. The blue arrows protruding from the eNodeBs in this screenshot represent the packets sent to UEs from corresponding eNodeBs at the time of the screenshot. If you were to play this NetAnim simulation you would see the UEs traverse their paths over the 285 second duration of the simulation.

Figure 3.2

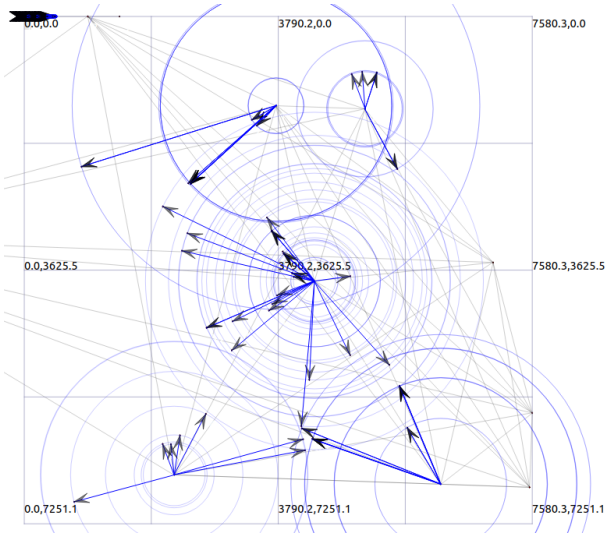

Figure 3.3

Figure 3.4: Comparison of Scenario 3 modelled in SUMO vs imported model in ns-3

3.3 Simulation Configuration

It was decided to use the ns-3 LENA module to simulate an LTE network. Attempting to run the simulations on a real-life LTE network over highways in Ireland was briefly considered. Inspiration to consider doing this came as a result of reading [34] which compared the performance of TCP CUBIC against TCP BBR over an 8 hour driving test on a Tier 1 LTE network in New York. However for practical reasons such as time constraints, not being able to drive and lack of test equipment this idea was scrapped. Simulation over ns-3 has many benefits over real-world applications such as allowing for more configurability of the parameters in my simulations. Simulation using ns-3 also allowed for the conduct of reproducible experiments.

In the ns-3 simulations conducted for this study one can be confident that the only parameters that change between simulation runs are the TCP algorithms themselves. By seeding the random variables in use in the simulations. One can carry out several runs of the simulation per TCP algorithm and then average these runs in order to have a high degree of confidence in the findings. Seeding also allows one to maintain consistency across the TCP algorithms in question per simulation run.

Realism was achieved through the use of the ns-3 Attribute system [43] which allows a high degree of parameterisation of simulations. Parameters that were set in the simulations were set in reference to guidelines that represent commercial deployments of LTE architecture.

3.3.1 Configuration of Simulation parameters

This section will go into depth about how the simulated LTE network in the simulations was configured. The main objective when designing this simulation was to emulate a real LTE network as close as possible while maintaining reproducibility of experimental runs. The first parameter of the LTE model that was considered for configuration was:

Mac Scheduler

One of the main benefits of LTE over previous mobile networks is its ability to allocate and prioritise bandwidth over UEs. The MAC scheduler gives LTE this ability. The Mac Scheduler assigns bandwidth resources to UEs and decides on how uplink and downlink channels are to be used by eNBs and the UEs of a cell. A MAC scheduler typically uses one scheduling algorithm with many possible parameters such as Quality of Service information from the EPC core, direct communication between UEs regarding the strength of communication signal, buffer queue information from previous layers informing it of how much data is queued for transmission. The simulations were configured to use the Proportional Fair algorithm which allocates bandwidth resources to UEs according to Quality of Service information from the EPC core and total throughput. This scheduler was chosen because according [35] it is the preferred MAC scheduler in commercial networks and one of the priorities of this study was to emulate a real LTE network.

Pathloss Model and Fading model

In ns-3, Propagation loss models [36] calculate the receiving signal power by taking into account the power of the transmission signal and the positions of the receiving and transmitting antenna positions. Multiple propagation models can be in use at the one time in ns-3 and the final receiving signal power takes into account all of these models.

The main pathloss model chosen for the simulations was the FriisPropagationLossModel which is an ns-3 implementation of [38]. This model was chosen because it is intended to simulate propagation in free space in the far field region where far field region is defined as distance to transmit being greater than three times the wavelength of the signal being propagated. Given that LTE band 7 was in use, which has a frequency of between 2620 - 2690 for downlink this corresponds to a wavelength of 0.11992 metres. Signals in the simulations are travelling far greater than 3×0.11992 metres so this model seems appropriate to use.

In wireless communications fading is the change of the attenuation of a signal as a result of many potential variables. These variables can include time, geographical position, and radio frequency. This could translate to real life factors such as weather (rain), obstacles obstructing the path of the signal or mobility of a node in the space medium.

In order to model this ns-3's matlab script was used for producing fading model trace files and it was given the appropriate parameters for the simulation. This produced a fading model tailored to a scenario with a vehicular node moving at 120kmph and downlink transmission at 2620 Mhz. This fading model is composed of 20,000 samples which ns-3 randomly selects from to implement fading at fixed intervals during a simulation.

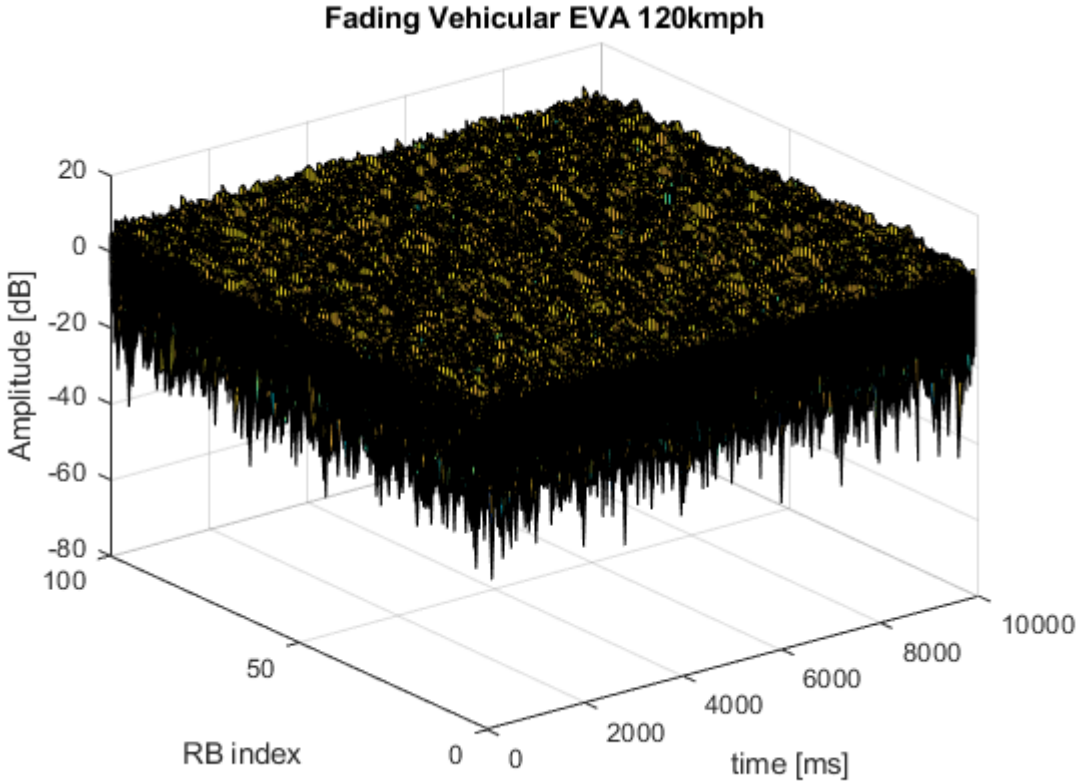

Figure 3.5: Excerpt of a custom generated fading trace for a vehicular scenario (speed of 120 kmph).

Transmission Power and Noise Level

In the interest of realism the most common transmission power for LTE devices, both UEs and eNodeBs was considered. [39] provides detailed guidelines on parameter values of LTE networks in simulation scenarios. From this the values for UE and eNodeb transmission power and noise level for a "Rural macro-cell" scenario were used. This uses 46dBm, 24dBm transmission power and 5dB, 7dB noise values for eNodeB and UE respectively.

RLC

In ns-3 the Radio Link Control layer supports three modes of operation. These are Acknowledged, Unacknowledged and Transparent. The acknowledged mode was used in the simulations since this is the most frequently used mode in commercial deployments. ns-3 equips the RLC layer with a DropTailQueue that acts as a transmission buffer. However the ns-3 implementation only gives a buffer size of 100 packets compared to a minimum of 750 packets found in real-life eNodeBs. ns-3's attribute Config system was used to change the buffer size to 750 packets.

LTE band and Resource blocks

The most common LTE bands in Europe are bands 1,3,7 and 8. Band 7 was chosen for this study. In order to configure this parameter the E-UTRA Absolute Radio Frequency Channel Number [40] (EARFCN) that corresponds to LTE band 7 was considered.

In order to assign 100 resource blocks per eNodeB which was the minimum in commercial deployments, a minimum bandwidth of 20Mhz was required. LTE band 7 has a bandwidth of 70Mhz which is more than enough for the 100 resource blocks that the eNodeBs were configured with.

Handover Algorithm

The algorithm used to determine when to trigger handover and pass the UE from the source eNodeB to the target eNodeB was the ns3::A2A4RsrqHandoverAlgorithm. This algorithm uses Reference Signal Received Quality (RSRQ) values to determine signal strength from UE to eNodeB. Handover occurs upon triggering of two events. The first of these events is triggered when the RSRQ between the serving eNodeB and UE becomes worse than a defined threshold. The second of these events is triggered when a neighbouring eNodeB's RSRQ is higher than the serving eNodeB's RSRQ by a certain offset. When the first and second events are triggered, the A2A4RsrqHandoverAlgorithm informs the eNodeB RRC to trigger a handover and UE connection is transffered from source eNodeB to target eNodeb.

3.3.2 Error model

The objective of this project is to evaluate how TCP variants perform over an LTE network. Four of these TCP variants interpret loss as congestion and loss is almost guaranteed in an LTE network given that it is a lossy wireless channel. Therefore it was deemed appropriate to add in an element of guaranteed loss in the simulations. This was achieved through the use of an ns3::RateErrorModel. This object was configured to model packet loss and set the rate of loss to 0.01 percent. An ns3::UniformRandomVariable was installed to distribute the loss that occurs around this 0.01 percent.

Finally the ns3::RateErrorModel was installed on the bottleneck link between the PGW node and the remote hosts which all traffic in the simulations flows through. This means that on top of other sources of loss in the simulations, such as from LTE lossy wireless channels, guaranteed loss will also occur as a result of this RateErrorModel.

Parameter	Parameter Value
Pathloss Model	FriisPropagationLossModel
Fading Model	EVA 120 kmph, (2620Mhz)
MAC Scheduler	Proportional Fair
eNodeB Transmission Power	46dBm
eNodeB Noise figure	5dB
UE Transmission Power	24dBm
UE Noise figure	7dB
RLC Mode	Acknowledged
RLC buffer queue	750 packets
LTE Band	7 (2620Mhz)
Resource Blocks	100

Table 3.1: Summary of simulation parameters

3.3.3 Modelling of Video Streaming in ns-3

Most simulation studies that compare TCP CCA performance do so through bulk sending of data at a constant rate. However in this work the focus was on carrying out a performance comparison of TCP CCAs with regards to video streaming applications. The nature of a real video stream is not constant. There is deviation in time between frames, the size of frames and as a result the actual rate of bytes in flight varies constantly.

[41] characterises traffic generated from a real Video on Demand service using Wireshark[42]. It focused on wireshark data regarding the types of frames being sent (I, P, B, and audio frames). They then used this wireshark data to model the video traffic using probability distribution functions. They used a logarithmic distribution to model the time between frames and a Weibull distribution to model change in size of frames in a video stream.

These probability distribution functions for time between frames and time to transmit frames were integrated into ns-3 via the ns-3 attribute system [43]. The application type used to transmit frames in ns-3 was an OnOffApplication. This application is parameterized through its ns-3 Attributes for "OffTime" and "OnTime". ns-3 defines "OffTime" as "A RandomVariableStream used to pick the duration of the 'Off' state" and "OnTime" as "A RandomVariableStream used to pick the duration of the 'On' state.".

			Video1		Video2		Video3	
			T (s)	S (Bytes)	T (s)	S (Bytes)	T (s)	S (Bytes)
Type of frames	I	PDF	Lognormal	Weibull	Logistic	Weibull		
		μ	0.4026	57177.5	0.3993	61379.2		
		Σ	0.0352	6592.3	0.0289	2910.3		
		KS	0.03	0.04	0.06	0.06	N/A	N/A
	P	PDF	Weibull	Lognormal		Weibull	Exponential	Weibull
		μ	0.1358	38446.8	0.1396	3447.4	0.1375	26923.8
		Σ	0.0688	9491.8	0.0972	12343	0.1261	14113.4
		KS	0.09	0.09	0.1	0.07	0.1	0.09
	B	PDF	Gamma	Weibull		Lognormal		Gamma
		μ	0.0508	12270.8	0.0514	10706.5	0.05	59314
		Σ	0.0338	7051.2	0.0559	8272.7	0.0911	3895.4
		KS	0.01	0.1	0.07	0.09	0.05	0.1
	Audio	PDF	Gamma	Logistic	Gamma	Logistic	Weibull	Normal
		μ	0.1484	2743.1	0.1468	2738.3	0.1457	2740
		Σ	0.0466	139.1	0.0614	137.6	0.0933	118.9
		KS	0.04	0.07	0.02	0.07	0.05	0.08

Figure 3.6: Probability distribution functions characterized by the mathematical behavior of the videos. This figure was taken from [41]

In order to replicate the mathematical behaviour of the real video streams found in [41], an "ns3::WeibullRandomVariable" object and an "ns3::LogNormalRandomVariable" object were parameterised according to the values found in Video1 of figure 3.6. The WeibullRandomVariable would feed values into "OnTime" implementing time for frames to transmit and the LogNormalRandomVariable would feed values into "OffTime" representing time between transmission of frames. These values mathematically represented the traffic from a 94 second, 5.2Mbps, mpeg2, 1920x1080 video stream. Thus the "DataRate" attribute of the OnOffApplication was set to 5000000bps or 5Mbps.

Parameter	Parameter Value
Remote	Ipv4Address of Test UE
DataRate	5000000bps or 5Mbps
PacketSize	1100 bytes
OffTime	ns3::LogNormalRandomVariable
OnTime	ns3::WeibullRandomVariable
MaxBytes	Unlimited bytes to transfer (UintegerValue (0))

Table 3.2: Summary of OnOffApplication parameters

4 Results

The main goal of this project was to evaluate how the chosen TCP variants performed over an LTE network with limited resources e.g. bandwidth. Performance was evaluated by considering goodput, change in CWND and delay/how these algorithms impact queues in a network. In order to evaluate this, this chapter goes into detail on the results outputted from the simulations. A total of three simulation scenarios were considered each performing an analysis of the chosen TCP variants over different network topologies and traffic loads. This chapter presents the results obtained from each simulation and the insights drawn from these results.

For all of the simulations each was repeated five times per TCP variant with different random number generator seeds. Random variable generators were used throughout the simulations. As already mentioned, in section 3.3.3 the Weibull and Logarithmic generators were used to produce frame-on and frame-off times. There are also many random variables within the ns-3 objects that were instantiated. Seeding these random variable generators allows for the repetition of simulations multiple times and averages out the results in an attempt to eliminate outliers and present results that are representative of the general use case for each TCP variant. The ns3::SeedManager::SetSeed function was used for initial seeding of the random variables and the SeedManager::SetRun function in order to maintain seed runs across TCP variants. This means each TCP variant receives the same seeding across their individual five runs per simulation.

Upon inspection of the simulation data after five runs it was concluded that the results between runs per TCP variant were similar enough to not warrant further runs. In the case of each TCP variant considered in this chapter the goodput, CWND and RTT were averaged over each variant's five runs respectively.

4.1 Scenario 1: Baseline Network

For the initial simulations the objective was to give the reader an idea of expected performance among the selected TCP variants under baseline conditions. These conditions are similar to those found in the first simulation presented by [29] which was discussed in detail in section 2.5. What is meant by baseline conditions is that they are composed of a single UE at a fixed distance from the eNodeB (usually 400-700 metres) with little to no background traffic. Therefore the Baseline Network simulation was composed of a single UE at a fixed 600 metres from a single eNodeB. There is no background traffic present in this simulation. The bottleneck bandwidth link between the LTE PGW node and the remote host that is generating the traffic has been given a capacity that far exceeds that which is required by the remote host node. The data rate of this link is 243Mbps and the transmission rate of the remote host emulating video streaming is only 5Mbps when sending. Thus it could be concluded that even if one knew nothing about each TCP variant that they should be able to utilize the excessive bandwidth resources to transmit video at 5Mbps.

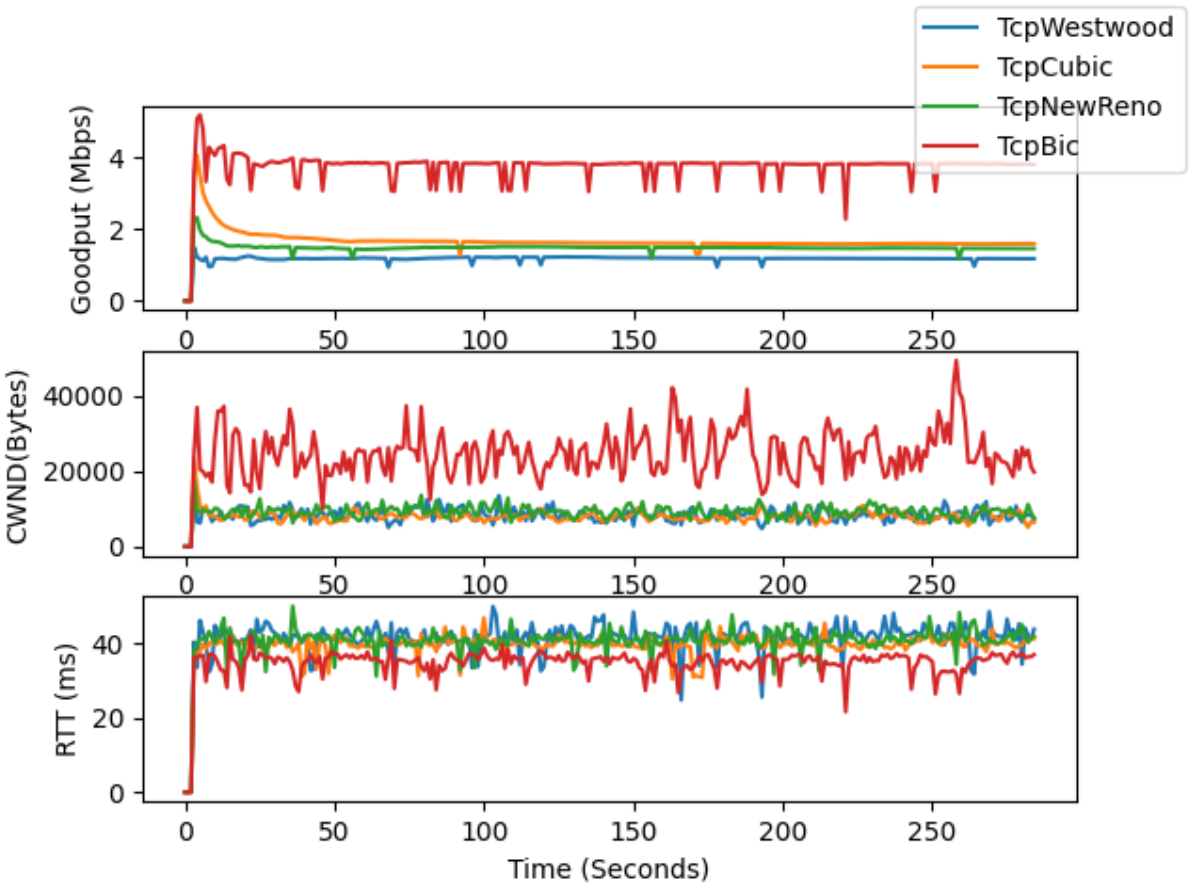

Figure 4.1: Results of loss based algorithms in Scenario 1.

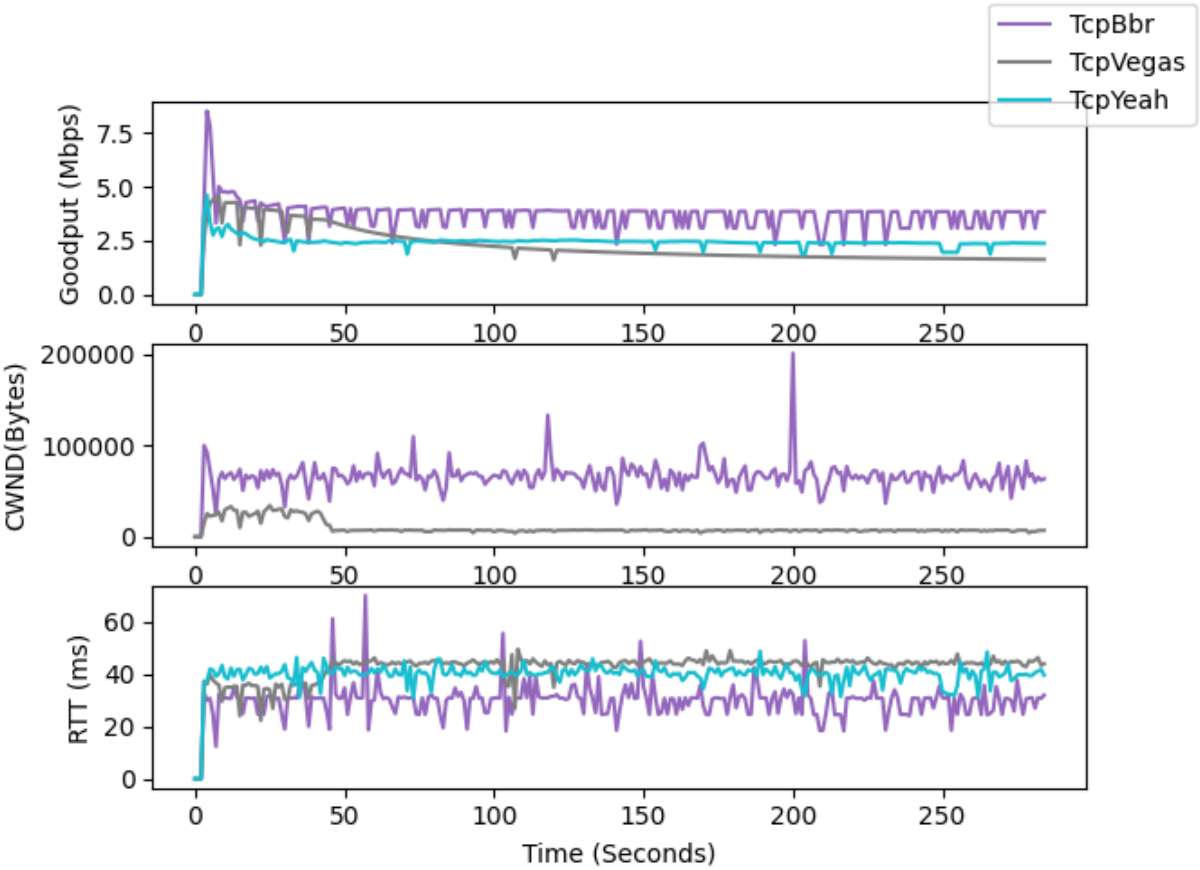

Figure 4.2: Results of delay/hybrid based algorithms in Scenario 1.

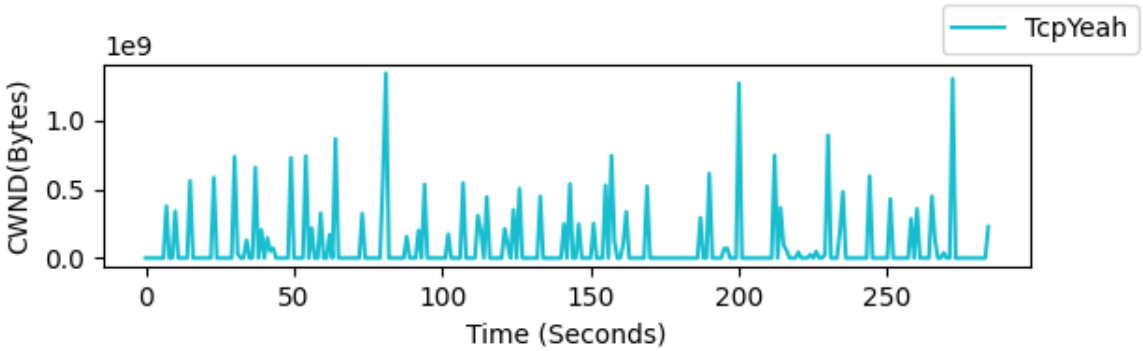

Figure 4.3: TCP Yeah CWND results for Scenario 1.

4.1.1 Scenario 1: Observations and reflection on results

In this scenario there was no lack of bandwidth or small queues in proportion to the traffic on the network. The algorithms had all the network resources to achieve a 5Mbps transmission rate. Loss should only have occurred from the pre-defined RateErrorModel loss of 0.01 percent and possibly as a result of operating over the wireless LTE channel. Realistically loss should not occur from dropped packets as links in this scenario have a high enough data rate to keep queues drained/stopped from forming in the first place.

Despite the available network infrastructure few of the TCP variants considered managed to utilise the available link and transmit at the full 5Mbps required for a high definition video stream. TCP Bbr and TCP BIC appear to be the only algorithms capable of utilizing these resources. This does not come as a surprise as these algorithms were designed with the intention of operating at a sending rate on the threshold of triggering packet loss or bottleneck link queuing respectively.

BIC achieves this with its binary chop search algorithm which remembers maximum values for CWND prior to a loss event and takes this into account in order to recover from loss quickly and ramp up to converge on utilizing available bandwidth. Similarly Bbr uses its BDP parameter to model the Bottleneck link. Bbr converges on this link's rate and uses its stages to monitor any changes in this links available bandwidth as opposed to interpreting loss as congestion. It should be noted that BBR achieves similar goodput to BIC while maintaining a lower delay, about 30ms vs 40ms respectively.

As previously noted, CUBIC is a less aggressive version of BIC. The polynomial function that captures the time since the last loss event in CUBIC allows it to take its CWND decisions over a time interval as opposed to explicitly at RTTs as is the case of BIC. This feature was added so that CUBIC would play fairly with other flows in the channel it operates over. However this refinement does appear to have reduced CUBIC's ability to fully utilise available bandwidth resources. Having said that it was still the second best performer out of the loss algorithms considered in this study.

In the case of the AIMD variants it was noted that Westwood differs from NewReno by the way it applies reduction to the CWND after a loss event. Westwood maintained lower goodput when comapared to NewReno. It appears that Westwood applied excessive reductions to its CWND after a loss event unlike NewReno which simply halves its CWND value. Westwood also appears to have created the most delay in the network.

Inspection of figure 4.9 shows how TCP Yeah switches between its "FAST" and "SLOW" modes. TCP Yeah's CWND values spike dramatically when in "FAST" mode under STCP rule, achieving peaks of 1×10^9 bytes or 909090 packets of size 1100 bytes.

It was concluded that the short lived nature of these spikes is due to TCP Yeah detecting large numbers of packets in the bottleneck queue link and switching to "SLOW" mode.

TCP Vegas begins its flow with relatively high goodput however it then is seen to be on a downwards trajectory, eventually becoming the worst out of the delay based algorithms. This is to be expected given its linear rate adaptation. What is surprising about the Vegas results is that they do not perform the best with regards to delay which was expected. Delay values begin low but around 50 seconds into the flow Vegas's delay values increase and remain this way for the rest of the flow.

It should be noted that all of the delay algorithms appear to have outperformed the loss based algorithms in this scenario bar TCP BIC.

4.1.2 Comparison of results

As was mentioned in section 2.5, [29] compared how the five most common TCP variants at the time utilized radio resources over LTE networks. [29]'s "Base behaviour" performed a similar simulation to Scenario 1 with a single UE at a fixed distance of 700m and no background traffic.

The results of Scenario 1 differ from the "Base behaviour" simulation of [29]. It should be noted that the TCP variants included for comparison are different for each simulation for example TCP BBR in this study. [29] managed to reach goodputs of 35Mbps with the same network topology as Scenario 1. The only change between these simulations was a 100 metre difference in distance from UE to eNodeB. When Scenario 1 was tested with a BulkSendApplication it was concluded that the capacity of the bottleneck LTE link at 600 metres from UE to eNodeB was around 5Mbps post initial transient phase. Goodputs of 35Mbps are theoretically achievable over LTE however, it is rare to achieve this in real-life. Scenario 1's bottleneck link is more representative of what is commonly achieved in real-life.

In Scenario 1 only one loss-based variant managed to reach this 5Mbps utilisation which was TCP BIC. TCP CUBIC, NewReno and Westwood all showed progressive decline in goodput values respectively. In the "Base behaviour" scenario of [29], all of the loss-based variants managed to reach full link utilisation (35Mbps). Given that we discussed the mechanisms for why certain TCP loss variants fail to utilise resources to the extent that others do in section 2.2 it is curious that all of the loss based variants in [29] reached full link utilisation post initial transient phase. This suggests a lack of loss in [29]'s simulations as these variants differ with how they react to loss. A lack of loss is unrealistic.

In [29] none of the delay-based variants managed to reach full link utilisation. The same can be said for Scenario 1 with the exception of TCP BBR.

[29] observed that loss-based variants created longer queuing delays in general when compared to the queuing delays of their delay-base counterparts. This can be seen to some extent in Scenario 1 were the delay based TCP BBR achieves the lowest queuing delay of the simulation with TCP Vegas and TCP Yeah matching if not having lower queuing delay values than their loss-based counterparts.

The differences in results between [29]'s "Base behaviour" simulation and this studies Scenario 1 are significant enough to confirm that this study was worth performing. [29] was released in 2016 and used ns-3.25 to perform its simulations. The accuracy of LTE simulations has been built on with every ns-3 release up to the current ns-3.30. It could be that the continued modelling of LTE in ns-3 over this time or the detail to which this studies simulations were parameterised is responsible for the difference in results acquired.

4.2 Scenario 2: Introducing background traffic

For this scenario the goal was to introduce background traffic on the network. Three new UE nodes were added. These were all placed at a fixed distance of 600 metres from one eNodeB. An ns3::BulkSendApplication was installed on each of the three new UEs. These applications simply try to fill up available bandwidth by generating as much traffic as possible. The remote host was configured to use TCP NewReno as the default CCA for the flows between the three new UEs and used whichever CCA was currently up for simulation on the flow between the test node UE. The link between the remote hosts and the PGW was given a potential data rate of 15Mbps. This data rate was used as TCP NewReno averaged about 1.7Mbps in Scenario 1. Given that there are three NewReno flows in this scenario and a desired 5Mbps goodput on our video stream flow that comes to 10MBps. This sets the target total load of our network to 66 percent which was considered sufficient to produce moderate congestion.

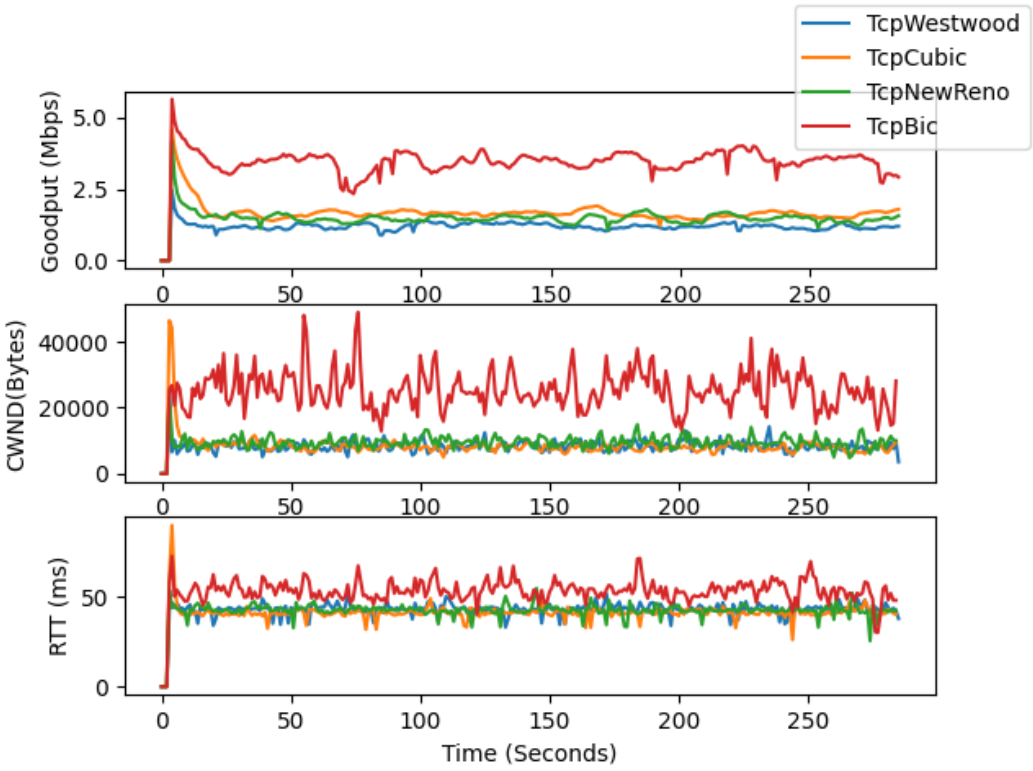

Figure 4.4: Results of loss based algorithms in Scenario 2.

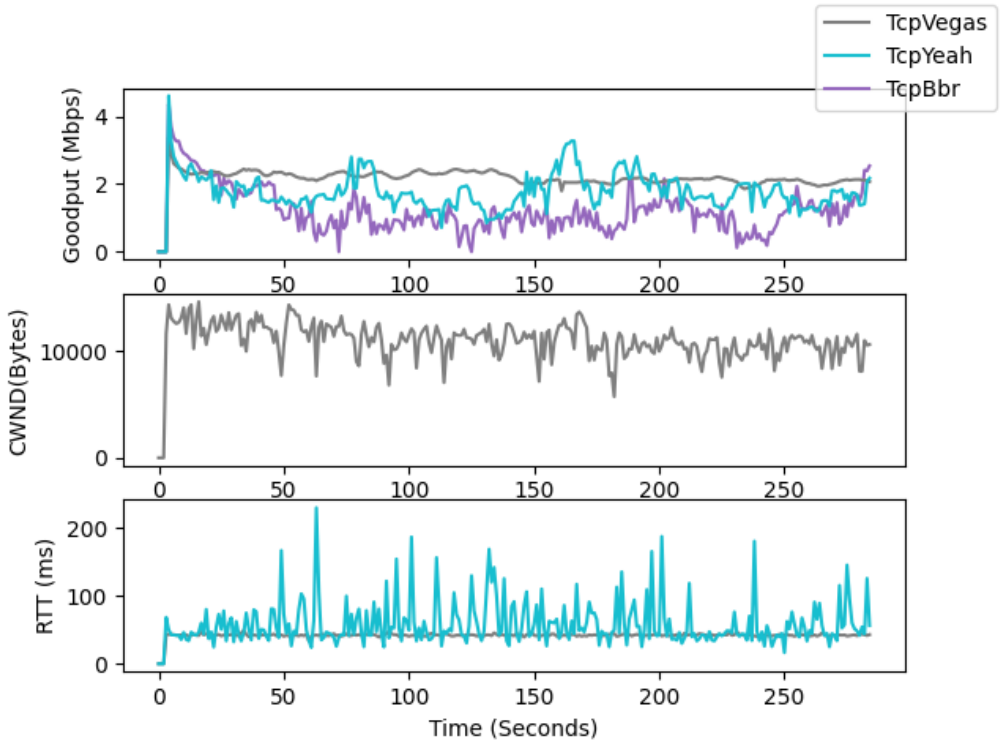

Figure 4.5: Results of delay/hybrid based algorithms in Scenario 2.

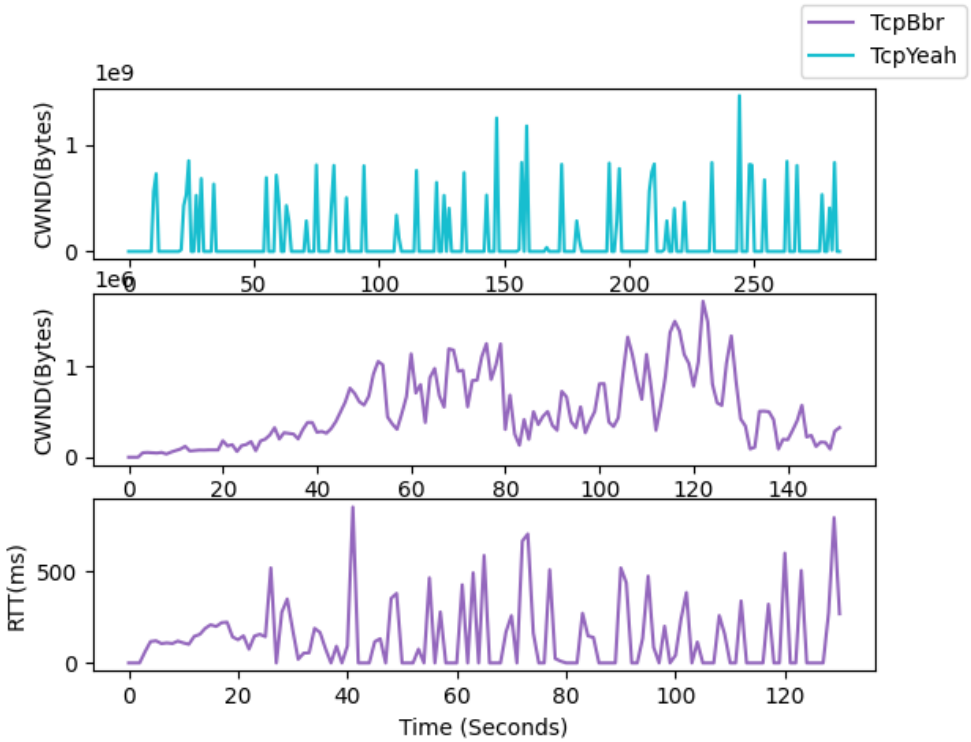

Figure 4.6: CWND of TCP Yeah, CWND and delay of TCP BBR results for Scenario 2.

4.2.1 Scenario 2: Observations and reflections on results

If one compares these results with the previous scenario the most obvious change is the overall increase in delay for all algorithms by at least 10ms. RTT values are now at a minimum of 50ms whereas in scenario 1 we had RTT as low as 30ms in the case of BBR.

Another change common amongst all variants in this scenario is the variable nature of the performance metrics. In scenario 1 relatively smooth goodput and stable CWND values established themselves after the initial transient phase. However in scenario 2 the variants are constantly adapting to changes in the available bandwidth caused by competing TCP flows.

TCP BIC went from having the lowest delay values to having the highest delay values of the loss based algorithms. BIC is known to be aggressive with regards to increasing its sending rate. For the majority of its operation time the nature of how BIC increases its sending rate is exponential. BIC is also RTT synchronized meaning it rarely takes other TCP flows in the channel into account. These factors make BIC aggressive allowing it to use the channel more efficiently but produce larger queuing delays as a consequence.

When looking at the delay based algorithms it can be seen that TCP Vegas manages to achieve similar, if not higher, goodput than in scenario 1. It also did not cause queuing delay spikes like its delay based counterparts Yeah and Bbr.

TCP Yeah spikes its CWND values again however this time it also appears to have negatively affected queuing delay. Bbr went from our most convincing performer to performing poorly with regards to spiking queuing delay and delivering very inconsistent goodput.

4.3 Scenario 3: Introducing trace mobility and high background traffic.

This scenario was the capstone for this project. This scenario involved a total of nine eNodeBs and fifty-one UEs. These UEs are no longer at a fixed distance from their eNodeBs. They have a mobility model that uses the imported SUMO trace file of the section of M3 motorway that was referenced previously. This SUMO trace file specifies the position, velocity and acceleration of every UE for the duration of this scenario. The test node from which we take our performance statistics will begin traversing the M3 motorway segment at the beginning of the simulation and end the simulation at the other end of the motorway segment. The eNodeBs are at fixed locations. Their locations correspond to existing locations of masts around the same section of M3 motorway that the UEs locations are imported from. All locations have been converted from real-life latitude longitude co-ordinates to cartesian co-ordinates.

This scenario also sees the introduction of handover. Handover, as discussed before, is the process of transferring UE connection from one eNodeB to another. This was not necessary in previous simulations as UEs were at fixed locations and only a single eNodeB was present. In this simulation handover occurs automatically via the X1 handover interface and is determined to occur via the A2A4RsrqHandoverAlgorithm.

Only the flow from remote host to test node UE is operating over TCP, the other UEs are receiving UDP traffic that is transmitted at roughly 2.5Mbps each. The sole purpose of this UDP traffic is to generate background traffic on the network. This gives a total network traffic of 130Mbps. The link was set between the PGW and remote hosts to be 260Mbps in an attempt to set the target total load of this network to 50 percent. Given the complexity of this network it is likely that bottleneck links will occur elsewhere. This is the only scenario in which the fading model is active as it is the only scenario in which UE nodes are moving.

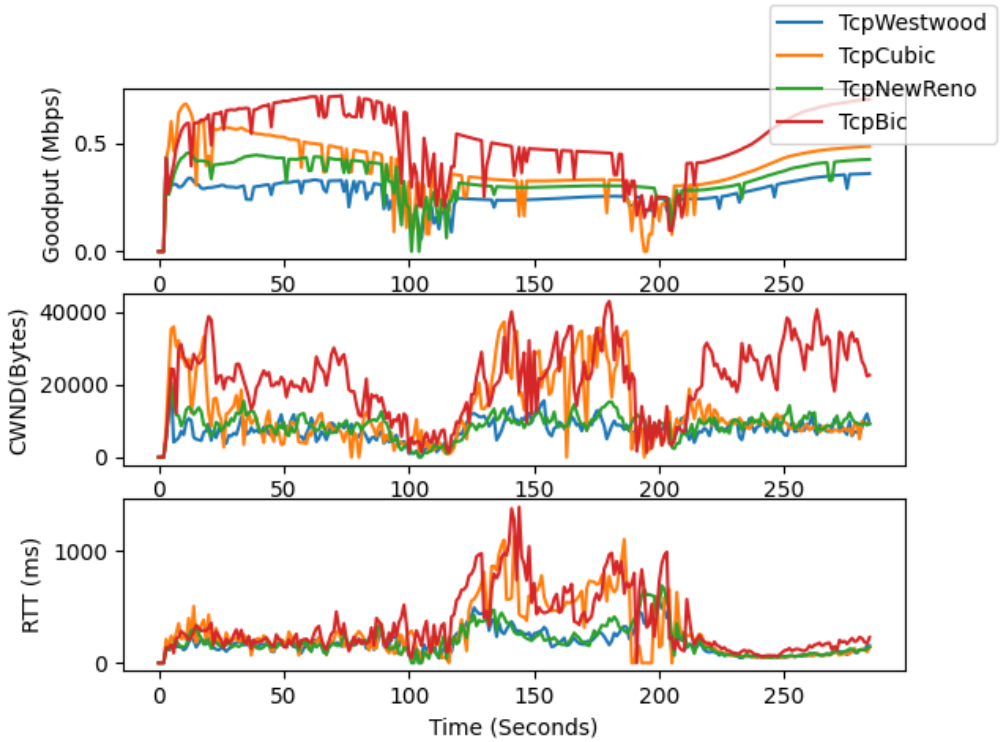

Figure 4.7: Results of loss based algorithms in Scenario 3.

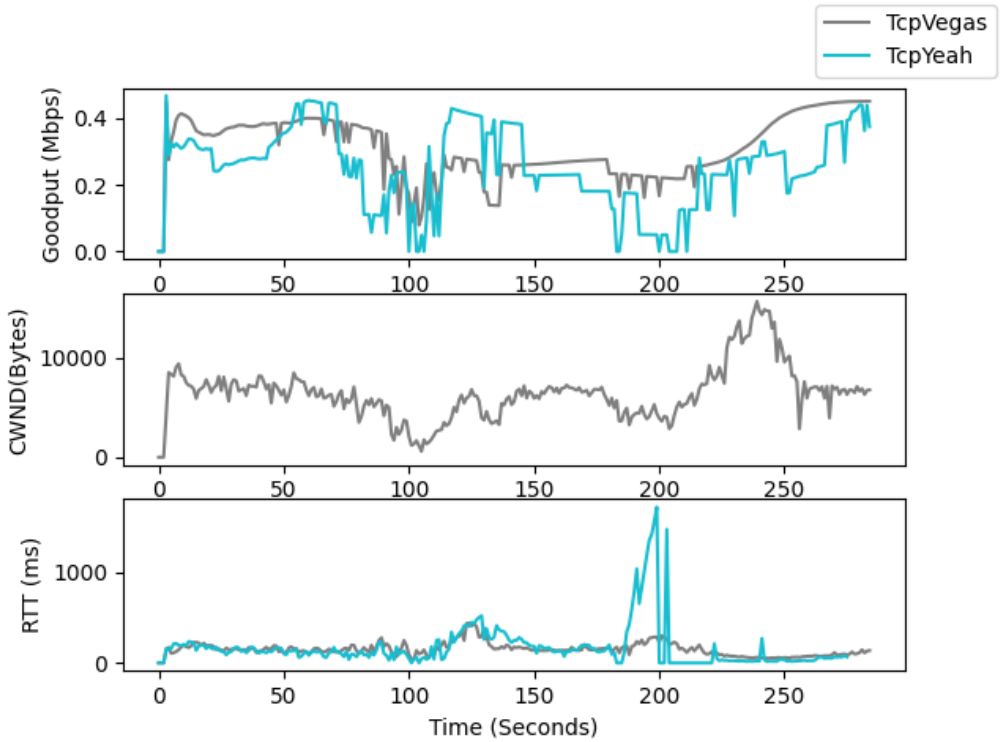

Figure 4.8: Results of delay/hybrid based algorithms in Scenario 3.

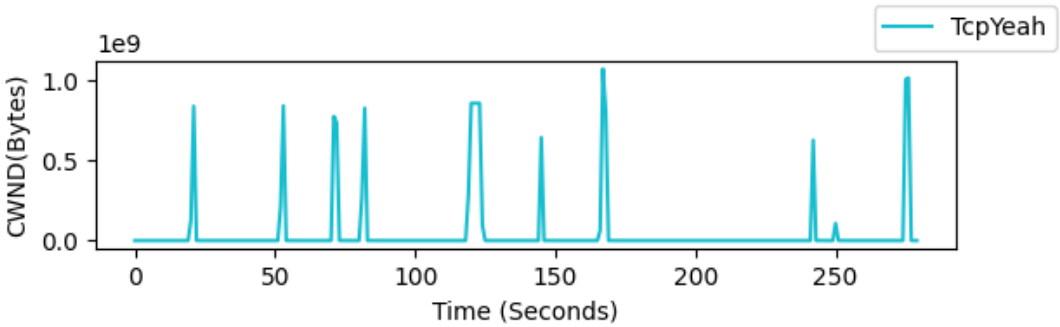

Figure 4.9: TCP Yeah CWND results for Scenario 3.

4.3.1 Scenario 3: Observations and reflections on results

The first observation one is likely to make when looking at these results is that there are 2 sectors of distinct poor performance that are common amongst all the variants. These sectors begin at around 100 seconds and 190 seconds into the simulation respectively. These sectors in the graph represent real-life areas of poor coverage on the segment of M3 motorway the scenario is modelled around. In the case of the 100 second outage the test node UE is located between eNodeBs at cartesian co-ordinates 2236.78,6849.12 and 4329.76,3952.77. These co-ordinates represent the location of real-life LTE masts and can be reverse calculated into latitude, longitude. A very similar event occurs at 190 seconds.

Originally when this scenario was designed it modelled the real-life scenario too well. The complete outage that was experienced in real life when streaming video and commuting also occurred in this simulation. The original results produced a complete outage with zero goodput across all variants between 100 seconds and 200 seconds because there was no coverage at all on this section of motorway. For the purpose of TCP variant comparison and producing interesting results an eNodeb was added at 4329.76,3952.77 to introduce coverage in this area. This mast is at the mid-point of the masts that the test UE is connected to at 100 seconds and 220 seconds.

The second observation one might make when looking at these results is the general drop in goodput when compared to the other scenarios. This scenario includes fading traces that model signal loss as a result of the constant movement of the UEs at high speeds. Low goodput can also be explained by the distance of the test node from eNodeBs in this scenario. The distance from test node to eNodeB is constantly changing in this scenario however an average distance of about 2 kilometres was estimated. Low goodput could also be attributed to the large amount of traffic that is distributed across this network as mentioned in the introduction to this scenario.

Despite the common occurrence of low goodput one can still see a clear differentiation between TCP variants. Once again it can be observed that TCP BIC aggressively outperform the other loss based variants. Consistently achieving higher goodput over the course of the scenario.

TCP CUBIC also appears to have outperformed Westwood and NewReno to a greater extent than in previous scenarios. Again the exponential nature of the binary chop search algorithm allows CUBIC and BIC to ramp up their CWNDs after a loss event and utilize available bandwidth faster than these other loss variants.

When looking at the loss based variants we see a reduction in the number of CWND spikes occurring for TCP Yeah. This is likely due to TCP Yeah less frequently deeming the queues small enough to go into FAST mode. Again we see TCP Vegas achieving very solid performance. Vegas maintains respectable goodput while keeping queuing delay consistently low. We also see Vegas's CWND being adapted to available bandwidth resources over the course of the simulation, barely experiencing a dip in performance at 200 seconds and rising to claim resources from 210 seconds and beyond.

Unfortunately BBR results are not present for this scenario. When it was attempted to run five runs for BBR and the furthest a single run got was 25 seconds. After this no performance metrics were outputted. It is not believed that these BBR results are representative of BBRs performance in real life implementations. It is suspected that integrating a BBR written for ns-3.27 into ns-3.30 may be a cause of the problem. It is also known that the BBR version implemented by Claypool et al.[19] that was used in these simulations was never integrated into ns-3 despite being submitted to WSN32018. There were probably good reasons for this of which some of these reasons may be the cause of the poor BBR performance experienced in this study. From reading Claypool et al.[19] it is noted that they had several TODO elements with regards to their BBR implementation. These TODO elements are given in Figure 4.10.

TODO

Below are BBR features *not* currently supported by BBR' (section numbers below refer to CCYJ17):

- BBR' assumes that the application always has data to send, thus the transmission rate is limited by the congestion control algorithm, not the application. BBR includes features to handle transmissions that are application-limited. See Section 4.1.1.4. This also means BBR' does not support restarting a flow from idle, a special case in BBR. See Section 4.3.4.4.
- BBR' only transitions to/from PROBE_RTT from PROBE_BW, while BBR has additional transitions for PROBE_RTT. See Section 4.3.5.
- BBR' does not include the "send quantum" in BBR which is used to amortize per-packet host overheads involved in the sending process. The send quantum parameter can be helpful at low rates with small packets. See Section 4.2.2.

Figure 4.10: Features of BBR' that Claypool et al.[19] have yet to implement.

Any of these issues could have been responsible for BBR failing to perform in this scenario.

This chapter presented a graphical representation of the data outputted for each of the simulations conducted. It also provided detail on simulation setup and observations on the results obtained in each of the three scenarios considered. The following chapter will conclude this work and suggest some directions for further research.

5 Conclusion

In this dissertation it was have demonstrated how a variety of TCP algorithms perform when operating over an LTE network in a high mobility scenario with limited resources. In general loss based algorithms fail to reach full link utilisation due to their over-reaction to loss. The exception to this was TCP BIC which due to the aggressive nature of how it recovers from loss managed to maintain relatively high goodput although it did create large queuing delays as a result.

There is encouraging behaviour in the case of delay-based algorithms. By taking delay into account all of these algorithms attempt to model the link they operate over to some extent. Unlike loss-based algorithms whose cycles intentionally cause queue overflow in order to receive signals on how to proceed. Perhaps the most promising of these algorithms was TCP BBR which achieved full link utilisation while simultaneously keeping queues at the lowest values seen in Scenario 1.

One of the main intentions of this work was to answer the question "Which TCP variant is best suited to utilise the limited LTE resources that tend to be distributed around motorways and thus be best for video-streaming to autonomous vehicles in the future?". If the answer to this question was solely based off of the results produced in this study the answer would probably be TCP BIC. TCP BIC utilised the resources available to it and due to the nature by which LTE assigns fixed Resource blocks per UE, the effects of TCP BIC's aggression on concurrent flows should be limited over LTE.

An honorable mention should be given to TCP BBR. It is likely that had the Claypool et al.[19] implementation not behaved so strangely when traffic loads were introduced it could have utilised available resources similar to TCP BIC while maintaining low queuing delays which is the ultimate goal for a TCP congestion control algorithm.

This study justified its ability to answer the above question by re-creating the factors that affect operating a video-stream while progressing along a motorway such as limited LTE resources, signal fading at high speeds, handovers, brief network outages etc.

5.1 Future Work

This section will detail elements of this dissertation that one might investigate further if there was more time to do so. If future M.A.I students sought to carry on with the ideas explored in this project these are the areas they could pursue.

DCE module

TCP Bbr did not demonstrate the same characteristics that it does in real-life or in other people's experiments. It was not originally intended to use the Claypool et al.[19] implementation of this algorithm. In fact it was not originally intended to use any of the native ns-3 implementations of TCP CCAs. The original plan was to use the ns-3 DCE module [48] for TCP CCA implementations.

DCE or Direct Code Execution provides facilities to execute, within ns-3, existing implementations of userspace and kernelspace network protocols or applications without source code changes. The intention was to use DCE to install a compiled linux kernel stack onto the ns-3 nodes in the simulations. This would enable the use of real linux kernel implementations of TCP CCAs. These are the same implementations that are in use by linux based operating systems across the world in many different types of networks including LTE. This means they are heavily debugged and proven to work over a vast range of scenarios. They certainly would not be missing several key features that could affect performance as we saw in the case of the [19] BBR implementation. The use of these linux kernel CCAs would have cemented the results of the TCP CCAs obtained in the simulations, accurately representing their real-life performance over LTE. The use of DCE could also have allowed the use of a real video streaming application to stream video over the LTE network.

Multiple weeks were spent attempting to get the DCE module working. DCE is not supported in ns-3.30 so to begin with another version of ns-3 had to be installed. Dependency issues were ran into next, the last DCE version was released in January of 2018. The only resources to solve these issues were the ns-3 forum and the outdated DCE manual . DCE required an older version of glibc:GNU C Library. In order to get glibc Ubuntu 16.04 had to be dual booted. Running a VM was not an option as the simulations required a lot of processing power. Once DCE built successfully further problems appeared such as manually routing traffic in over the network. There was no documentation on this. Another problem was that applications to generate traffic would have to be written from scratch in C as ns3::Applications did not work with DCE. This meant all of the work done in order to achieve realistic modelling of video transmissions would not carry forward.

In an attempt to get advice on these issues an email was sent to researchers that had written research papers involving the use of DCE. Unfortunately due to time constraints by the time one of these researchers responded the use of DCE in this dissertation had been abandoned. If there was more time or if a suggestion was to be made for a future M.A.I student to continue with this study including DCE in their simulations would be recommended . The use of DCE would have added more realism and compounded the results of this study.

Video Streaming

The original plan for this study was to stream a video file over the LTE network in the simulations as opposed to emulating video stream traffic. This would then allow for the calculation of some Quality of Experience evaluation on the file received by the UE in the simulation. The plan was to compare these quality of service metrics across TCP algorithms and see if network performance statistics such as high throughput co-related with perceived QoE of the end user.

However, the lack of a suitable TCP video streaming module for ns-3 caused a pivot in the research plans. The most well known module for video streaming in ns-3 is Evalvid [49] which communicates strictly over UDP. This was obviously not suitable for a TCP based project.

Therefore one avenue for future research would be to explore how a real video stream could be integrated into the simulations and then perform QoE analysis on this video-stream. One possible way they could go about this is by using the DCE module previously discussed. DCE allows for execution of binaries from a linux kernel which is installed on an ns-3 node. A simple TCP video streaming application could be written in C and compiled to run on ns-3 nodes using DCE to achieve this.

Inclusion of experimental TCP CCAs

There was an intention to include experimental congestion control schemes for comparison in this study. Experimental algorithms are algorithms that have shown promise in simulations and research but are not commonly used in networks today. One example of these congestion control schemes is Sprout [18]. This congestion control scheme was specifically designed for cellular networks. Sprout uses stochastic forecasts to model the network it operates over based on packet arrival times.

Sprout consistently produces very promising results on Stanford's Pantheon [50]. Pantheon is self described as "a community evaluation platform for academic research on congestion control". It's essentially a test-bed of wired and cellular nodes that routinely runs performance comparisons on 17 different congestion-control schemes.

If one looks at Pantheon's results page and iterates through the pages of results until you find experiments done over "Cellular" channels you will see that Sprout consistently outperforms other congestion control schemes with regards to throughput and delay.

If there was more time one could include experimental congestion control schemes like Sprout in their LTE comparison simulations using ns-3. Therefore for anyone seeking to build on this project they could look at implementing SPROUT in ns-3. The source code can be found here.

Bibliography

[1] Sundaresan K., Park SJ., Sivakumar R. (2005) Transport Layer Protocols in Ad Hoc Networks. In: Mohapatra P., Krishnamurthy S.V. (eds) Ad Hoc Networks. Springer, Boston, MA

[2] Cisco, V.N.I., 2018. Cisco visual networking index: Forecast and trends, 2017–2022. White Paper, 1.

[3] Litman, T., 2017. Autonomous vehicle implementation predictions (p. 28). Victoria, Canada: Victoria Transport Policy Institute.

[4] Kurose, J.F. and Ross, K.W., Computer networking: a top-down approach (pp. 607967-5). Addison Wesley.

[5] Henderson, T.R., Lacage, M., Riley, G.F., Dowell, C. and Kopena, J., 2008. Network simulations with the ns-3 simulator. SIGCOMM demonstration, 14(14), p.527. https://www.nsnam.org/

[6] Krajzewicz, D., Hertkorn, G., Rössel, C. and Wagner, P., 2002. SUMO (Simulation of Urban MObility)-an open-source traffic simulation. In Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002) (pp. 183-187).

[7] Floyd, S., Henderson, T. and Gurtov, A., 1999. The NewReno modification to TCP's fast recovery algorithm.

[8] Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M.Y. and Wang, R., 2001, July. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In Proceedings of the 7th annual international conference on Mobile computing and networking (pp. 287-297).

[9] Xu, L., Harfoush, K. and Rhee, I., 2004, March. Binary increase congestion control (BIC) for fast long-distance networks. In IEEE INFOCOM 2004 (Vol. 4, pp. 2514-2524). IEEE.

[10] Ha, S., Rhee, I. and Xu, L., 2008. CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS operating systems review, 42(5), pp.64-74.

[11] Levasseur, B., Claypool, M. and Kinicki, R., 2014, May. A TCP CUBIC implementation in ns-3. In Proceedings of the 2014 Workshop on ns-3 (pp. 1-8).

[12] Brakmo, L.S. and Peterson, L.L., 1995. TCP Vegas: End to end congestion avoidance on a global Internet. IEEE Journal on selected Areas in communications, 13(8), pp.1465-1480.

[13] Baiocchi, A., Castellani, A.P. and Vacirca, F., 2007, February. YeAH-TCP: yet another highspeed TCP. In Proc. PFLDnet (Vol. 7, pp. 37-42).

[14] Kelly, T., 2003. Scalable TCP: Improving performance in highspeed wide area networks. ACM SIGCOMM computer communication Review, 33(2), pp.83-91.

[15] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H. and Jacobson, V., 2016. BBR: Congestion-based congestion control. Queue, 14(5), pp.20-53.

[16] Jain, V., Mittal, V. and Tahiliani, M.P., 2018, June. Design and implementation of TCP BBR in ns-3. In Proceedings of the 10th Workshop on ns-3 (pp. 16-22). https://github.com/Vivek-anand-jain/Reproduce-TCP-BBR-in-ns-3 https://www.nsnam.org/research/wns3/wns3-2018/accepted-papers/

[17] ComReg site viewer 2020, "Commission for Communications Regulation is the general communications regulator for Ireland", viewed 03 March 2020, https://siteviewer.comreg.ie/

[18] Winstein, K., Sivaraman, A. and Balakrishnan, H., 2013. Stochastic forecasts achieve high throughput and low delay over cellular networks. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13) (pp. 459-471).

[19] Claypool, M., Chung, J.W. and Li, F., 2018, June. BBR' an implementation of bottleneck bandwidth and round-trip time congestion control for ns-3. In Proceedings of the 10th Workshop on ns-3 (pp. 1-8). https://dl.acm.org/doi/abs/10.1145/3199902.3199903

[20] Scholz, D., Jaeger, B., Schwaighofer, L., Raumer, D., Geyer, F. and Carle, G., 2018, May. Towards a deeper understanding of tcp bbr congestion control. In 2018 IFIP Networking Conference (IFIP Networking) and Workshops (pp. 1-9). IEEE.

[21] Turkovic, B., Kuipers, F.A. and Uhlig, S., 2019. Fifty shades of congestion control: A performance and interactions evaluation. arXiv preprint arXiv:1903.03852.

[22] Jaeger, B., Scholz, D., Raumer, D., Geyer, F. and Carle, G., 2019. Reproducible measurements of TCP BBR congestion control. Computer Communications, 144, pp.31-43.

[23] 3GPP, "LTE Overview", 2008, last viewed on 12 March 2020. https://www.3gpp.org/technologies/keywords-acronyms/98-lte

[24] Hussien M., 2016, "LTE-EPC Technology Essentials - 3GPP Language Course", LinkedIn, viewed on 02 April 2020.

[25] yateBTS, "Evolved Packet Core (LTE EPC), Evolved Packet Core (LTE EPC), 2018, viewed on 02 April 2020. https://yatebts.com/solutions$_and_technology/lte-epc/$

[26] Nicola Baldo. "The NS-3 LTE module by the LENA project", 2011. viewed on 28 April 2020. https://www.nsnam.org/docs/models/html/lte-design.htmloverview

[27] Baldo, N., 2011. The ns-3 LTE module by the LENA project. Center Tecnologic de Telecomunicacions de Catalunya, viewed on 02 April 2020. https://www2.nsnam.org/tutorials/tutorials/consortium13/lte-tutorial.pdf

[28] Pillmann, J., Behnke, D., Sliwa, B., Priebe, M. and Wietfeld, C., 2018, August. Efficient and Reliable Car-to-Cloud Data Transfer Empowered by BBR-enabled Network Coding. In 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall) (pp. 1-5). IEEE.

[29] Atxutegi, E., Liberal, F., Grinnemo, K.J., Brunstrom, A., Arvidsson, Å. and Robert, R., 2016, July. TCP behaviour in LTE: Impact of flow start-up and mobility. In 2016 9th IFIP Wireless and Mobile Networking Conference (WMNC) (pp. 73-80). IEEE.

[30] ns-3 Built-in Mobility models 2019, "The mobility support in ns-3 includes:", viewed 18 January 2020, https://www.nsnam.org/docs/models/html/mobility.htmlmobilitymodel

[31] Arbabi, H. and Weigle, M.C., 2010, December. Highway mobility and vehicular ad-hoc networks in ns-3. In Proceedings of the 2010 Winter Simulation Conference (pp. 2991-3003). IEEE. http://dx.doi.org/10.1109/WSC.2010.5678993

[32] Kang, S.S., Chae, Y.E. and Yeon, S., 2017, August. VANET routing algorithm performance comparison using ns-3 and SUMO. In 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT) (pp. 1-5). IEEE. http://dx.doi.org/10.1109/CAIPT.2017.8320746

[33] Raj, C., Upadhayaya, U., Makwana, T. and Mahida, P., 2014. Simulation of VANET using ns-3 and SUMO. International Journal of Advanced Research in Computer Science and Software Engineering, 4(4).

[34] Li, F., Chung, J.W. and Jiang, X., 2017. Driving TCP congestion control algorithms on highway. Proceedings of Netdev, 2.

[35] yateBTS, "An introduction to the LTE MAC Scheduler", viewed 11 March 2020 https://yatebts.com/an-introduction-to-the-lte-mac-scheduler/

[36] ns-3, "The ns-3 propagation module defines two generic interfaces...", viewed 11 March 2020 https://www.nsnam.org/docs/models/html/propagation.htmlpropagationdelaymodel

[37] G. Piro, N. Baldo. M. Miozzo, "An LTE module for the ns-3 network simulator", in Proc. of Wns3 2011 (in confloatjunction with SimuTOOLS 2011), March 2011, Barcelona (Spain)

[38] Friis, H.T., "A Note on a Simple Transmission Formula," Proceedings of the IRE , vol.34, no.5, pp.254,256, May 1946

[39] International Telecommunication Union, 2009, "Guidelines for evaluation of radio interface technologies for IMT-Advanced", viewed 11 March 2020 https://www.itu.int/pub/R-REP-M.2135

[40] Association of Radio Industries and Businesses, "Downlink E-UTRA Absolute Radio Frequency Channel Number (EARFCN)", as per 3GPP 36.101 Section 5.7.3., viewd on 11 March 2020 https://www.arib.or.jp/english/html/overview/doc/STD-T104v2$_1$0/5$_A$$ppendix/Rel11/36/36101-b50.pdf$

[41] Campo-Muñoz, W.Y., Astaiza-Hoyos, E. and Muñoz-Sanabria, L.F., 2017. Traffic modelling of the video-on-demand service through NS-3. Dyna, 84(202), pp.55-64.

[42] Wireshark, "Wireshark is the world's foremost and widely-used network protocol analyzer", viewed 18 March 2020. https://www.wireshark.org/

[43] ns-3, "Configuration and Attributes: In ns-3 simulations, there are two main aspects to configuration", viewed 02 April 2020. https://www.nsnam.org/docs/manual/html/attributes.html

[44] Levasseur, B., Claypool, M. and Kinicki, R., 2014, May. A TCP CUBIC implementation in ns-3. In Proceedings of the 2014 Workshop on ns-3 (pp. 1-8).

[45] Levasseur, B., Claypool, M. and Kinicki, R., 2014, May. CUBIC for ns-3, "An implementation of CUBIC in ns-3, designed based on the current literature", viewed on 19 March 2020. http://perform.wpi.edu/downloads/cubic

[46] Jain, V., Mittal, V. and Tahiliani, M.P., 2018, June. Design and implementation of TCP BBR in ns-3. In Proceedings of the 10th Workshop on ns-3 (pp. 16-22).

[47] Jain, V., Mittal, V. and Tahiliani, M.P., 2018, June. "Reproduce the results of TCP BBR", viewed on 19 March 2020. https://github.com/Vivek-anand-jain/Reproduce-TCP-BBR-in-ns-3

[48] Tazaki, H., Urbani, F. and Turletti, T., 2013, March. DCE Cradle: Simulate network protocols with real stacks.

[49] GERCOM. "NS-3 Evalvid module by GERCOM", 2013. https://gitlab.com/gercom/evalvid-ns3

[50] Yan, F.Y., Ma, J., Hill, G.D., Raghavan, D., Wahby, R.S., Levis, P. and Winstein, K., 2018. Pantheon: the training ground for Internet congestion-control research. In 2018 USENIX Annual Technical Conference (USENIXATC 18) (pp. 731-743).